UNIVERSIDAD POLITÉCNICA
DE MADRID

FACULTAD DE INFORMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE –
EUROPEAN MASTER ON SOFTWARE ENGINEERING

Adding Usability Mechanisms into Agile Requirements: Tool Support with U-Kunagi

Master Thesis

Luis Carlos Chacón Salas

Madrid, July 2013

This thesis is submitted to the Facultad de Informática at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of Master of Science on Software Engineering.

Master Thesis
Master Universitario en Ingeniería del Software – European Master on Software Engineering
Thesis Title: Adding Usability Mechanisms into Agile Methodologies: Tool Support with U-Kunagi
Thesis no: EMSE-2013-06
July 2013

Author: Luis Carlos Chacón Salas
Bachelor on Computer Science and Informatics
Universidad de Costa Rica

Supervisor:

Ana María Moreno
Ph.D. in Computer Science
Universidad Politécnica de Madrid

Departamento de Lenguajes y Sistemas
Informaticos e Ingeniería de Software
Facultad de Informática
Universidad Politécnica de Madrid

Co-supervisor:

Barbara Russo
Ph.D. in Mathematics
Università degli Studi di Trento

Computer Science Department
Faculty of Computer Science
Libera Univesità di Bolzano

Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte (Madrid)
Spain

ABSTRACT

The growing interest for integrating agile methodologies and usability has brought various challenges to practitioners. This research focuses on a specific part of these challenges that is related to the integration of usability mechanisms (features such as cancel, undo, warning, etc.) into agile requirements, usually written in the form of user stories. For this aim, a framework has been developed, conformed first by a well-defined modeling language that aims to formalize previous empirical research in the field, models of the impact of usability mechanisms into user stories, and a tool to help practitioners applying them to user stories. Results show that the use of this framework helps agile developers to think about usability from the beginning of the development process, without needing to be an expert in the subject. Our proposal can therefore complement other usability practices to improve the quality of use of software developed using agile methodologies.

TABLE OF CONTENTS

1. Introduction 1
2. Research Scope 3
2.1 Agile Requirements 3
2.2 Usability Mechanisms 4
3. Related Work 6
4. Objectives 8
5. Methodology 10
6. Preliminary Work 13
7. Usability Mechanisms Metamodel 15
8. Usability Mechanisms Profile 17
9. Usability Mechanism Models 21
9.1 Definition of the Models 21
9.2 Advantages of Using the UML Profile 24
9.3 Definition of Instances 27
10. Tool support 29
10.1 Tool Aim 29
10.2 Tool Selection Process 29
10.3 Description Of U-Kunagi 30
10.4 Validation 34
11. Conclusions 38
12. Acknowledgements 40
13. References 41
Appendix A. OCL constraint model for the UML Profile. 44
Appendix B. Usability Mechanism Models 48
Appendix C. Questionnaire for Validation. 56

LIST OF FIGURES

Figure 1. Activity Diagram of objectives 9
Figure 2. Metamodel 16
Figure 3. UML Profile described as UML package 18
Figure 4. Warning model 25
Figure 5. Warning model without UML Profile 26
Figure 6. Instance of user story 27
Figure 7. Instance of user story with warning usability mechanism 28
Figure 8. Acceptance Criteria Management 31
Figure 9. Instance of a User story in kunagi 31
Figure 10. Overview of usability mechanism addition 32
Figure 11. FInal instance of the user story in kunagi 33
Figure 12. Usability user story: Implementation of warning window 33
Figure 13. User Story with warning usability mechanism 36
Figure 14. User Story with Go Back, System Status and text entry Usability Mechanisms 36
Figure 15. Abort Command model 48
Figure 16. Abort Operation model 49
Figure 17. Favourties model 49
Figure 18. Go back model 50
Figure 19. Help model 50
Figure 20. Long Action model 51
Figure 21. Preferences model 51
Figure 22. Step by Step model 52
Figure 23. System Status model 52
Figure 24.Text Entry model 53
Figure 25. Undo model 54
Figure 26. Undo Reset model 55
Figure 27. Warning model 55

LIST OF TABLES

Table 1. Overview of Usability Mechanisms 4
Table 2. Sprints and User Stories of the project 12
Table 3. Mapping between usability mechanisms and actions [30] 13
Table 4. Definition of Warning Usability Mechanism [13] 23
Table 5. Summary of Tool Selection analysis 30
Table 6. Product backlog of one of the projects used for validation [35] 35

1. INTRODUCTION

Agile methodologies have become popular in the software development industry and have changed the way of approaching the development process. The agile approach aims to create software that is more aligned to the customer requirements and thus deliver quality in a faster pace when compared to traditional development. This close cooperation with the costumer is often seen as a way to deliver quality software, but, as explained by Jokela and Abrahamsson [1], this is not always the case, at least when taking in consideration the usability quality factor.

Usability is defined by Constantine and Lockwood [2] as the ease of use of software which comprises the ease of learning how to use it, the ease of using it efficiently, the ease of remembering from one use to the next and that it gives satisfaction to the users. Because of this, usability is an important quality factor, as it is found in several software quality classifications [3, 4, 5].

When building software for usability, the regular approach is to spend a considerable effort in analyzing and designing before the development phases [2]. This creates a sort of contradiction with the agile approach in which on each iteration a subset of features is chosen, analyzed, developed and delivered. This, essentially, because traditional methods for addressing usability require upfront allocation for designing and developing User Interfaces (UIs) and other usability related tasks [6]. Another issue is that usually neither developers nor customers are User Experience (UX) experts, so dealing with usability during the software development process is usually done by intuition [1].

However there are also similarities between agile methodologies and UX design. For example, both are costumer centered, UX design focuses on developing software for improving the user experience, with the user in mind, while agile methodologies aim to shorten the feedback loop between customers and developers [7]. Also both UX design and agile methodologies aim for quality, so a combination of both could rapidly create software that is useful and usable for the costumer.

There is a growing interest in the combination of agile methodologies and UX design and there are still various challenges that developers face when addressing this integration [8]. Traditional approaches to usability state that it impacts User Interfaces (UI) (for example [9, 10]) and the development process (for example [11, 12]). Juristo et al. [13] suggest considering usability features as functional requirements rather than as non-functional requirements. Their results show that this brings significant improvements in the

identification of the usability details needed for the project. Juristo et al. [14] consider that adding usability features may impact also the design of the software system. According to their results in a set of case studies, the addition of usability features into a software system will lead to significant changes on software design. We aim for considering these usability features and their impact on software design into the agile methodologies.

We focus our research on the specific problem of adding common usability mechanisms (such as cancel, undo, warning, etc.) to an application that is being developed and how these mechanisms interact with the usual agile requirements. Our main interest is twofold; on one hand, to explore the impact of integrating usability mechanisms into the agile requirements; and secondly to create a framework to help practitioners during this integration. This framework must support a simple process that non UX design experts can follow and must be flexible enough to face new technologies and usability features.

For fulfilling this goal, we start by defining the scope of this research into the agile requirements and into the usability quality attribute explaining what the usability mechanisms are. Then we show an overview of related work and how our objectives complement and extend it, on Section 3. We describe the objectives of this research on Section 4. On Section 5 we explain the methodology used to organize the work and acquire the results. We make an overview over the authors' work on which this research is based on Section 6. On Section 7, we present a metamodel that will be the base of a well-defined modeling language which is the base of the framework. This language is created on the form of a Unified Modeling Language (UML) profile and it is presented on Section 8. On Section 9 we discuss the use of the profile and instances of it for representing and working with different usability features. On Section 10 we present a tool that allows the automatic use of these instances inside an agile development process, also we present an assessment of the use of this tool. Finally, our conclusions are presented on Section 11.

2. RESEARCH SCOPE

2.1 AGILE REQUIREMENTS

In agile methodologies requirements are often defined in the form of user stories. A user story is a slim and high level way of representing functionality that is valuable to the end user of the application [15]. According to Cohn [15] user stories are composed of three aspects:

- A written description of the story used for planning and as a reminder
- Conversations about the story that serve to flesh out the details of the story
- Tests that convey and document details and that can be used to determine when a story is complete

A typical written description of a user story can be the following:

> *User Story I: As a user I want to be able to delete files*

With the description of the user story, acceptance criteria (or tests) are also written. These acceptance criteria will verify that the user story is developed such that it works exactly as the costumer expects it to work [15]. Writing acceptance criteria early helps to perceive the user assumptions and expectations before starting to work on the user story implementation [15]. An example of acceptance criteria for *User Story I* can be the following:

> *Acceptance Criteria I: Check that after the user clicks the GUI to delete, the file is deleted from the file system*

The details of the user story, and the conversations about it, are used to disaggregate the user story into its constituent tasks [15]. These tasks describe the set of activities that the developers have to do in order to implement the user story. This means that tasks are written in a more technical way. When the user story is split into tasks, the developers accept responsibility for each task [15]. An example of the task disaggregation for *User Story I* can be the following:

> *Task I: Create the GUI*
>
> *Task II: Implement the action that deletes the file*

The framework that we define in this research affects the treatment of agile requirements. This because the addition of usability mechanisms, as explained in the following sections,

affects user stories by modifying or creating its tasks and its acceptance criteria and by relating them with other user stories.

2.2 USABILITY MECHANISMS

Literature about UX design has identified different usability practices that improve effectively the quality of use of software systems [16]. Among these practices we find particular recommendations like giving the user the option to cancel an ongoing process [17, 18, 19], to undo a task [20, 21], giving feedback on what is going on in the system [20, 22], adapting the software to an user profile [23], providing clear and marked exits for the application [22], etc. These recommendations named as Usability Mechanisms [13] represent particular functionalities that should be treated as functional requirements with clear design implications [24]. Therefore in an agile approximation agile requirements and subsequent design will also be affected by them.

An overview of the Usability Mechanisms addressed in this research is shown on Table 1 [13].

Usability Mechanism	**Description**
System Status	To inform users about the internal status of the system
Warning	To inform users of any action with important consequences
Long Action	To inform users that the system is processing an action that will take some time to complete
Go Back	To go back to a particular state in a command execution sequence
Text Entry	To help prevent the user from making data input errors
Step by Step	To help users to do tasks that require different steps with user input and correct such input
Preferences	To record each user's options for using system functions
Favorites	To record certain places of interest for the user
Help	To provide different help levels for different users
Undo	To undo system actions at several levels
Undo reset	To undo several actions on an object
Abort command	To cancel the execution of a task in progress
Abort operation	To cancel the execution of an action or the whole application

TABLE 1. OVERVIEW OF USABILITY MECHANISMS

Previous authors have packaged a set of guidelines that empowers developers to know how the addition of a usability feature affects the analysis and design artifacts in the

development process [24]. We will work on the incorporation of such guidelines in agile requirements, and in particular in user stories.

3. RELATED WORK

This section discusses other works that have addressed the integration of usability practices in the agile domain. We used as starting point the systematic review performed by Silva da Silva et al. [7] to find specific papers that work on the inclusion of usability into agile requirements. These papers follow some approaches that have certain similarities to our research, so we studied them and in this section we summarize their most important findings and compare them to our research.

Düchting et al. [25] analyze how User Centered Requirements are considered in Scrum and Extreme Programing (XP). It concludes that, in their current forms, these agile methodologies have significant deficiencies when handling User Centered Requirements [25]. It finally recommends capturing User Centered Requirements in the Product Backlog, in the form of user stories.

Singh [26] finds some problems of using UX design within Scrum, for example, the lack of artifact development before the development starts, such as prototypes, and the difficulties of prioritizing usability requirements as compared to functional requirements. In order to try to avoid these difficulties, the authors propose U-SCRUM which includes some modifications to the SCRUM process. One of these modifications that is specially related to this research is the inclusion of personas in the creation of use stories. Personas are profiles that reflect the desired user experience with the product [26]. These personas then are cited in the user stories and are input for usability related acceptance criteria.

Beyer et al. [27] recommend a new process in order to combine UX Design with agile methodologies. This process assumes UI experts inside the development team. The steps of the process recommend doing contextual inquiries about UI issues to different potential users, and then this information, in the form of diagrams and mockups, will be used to generate the user stories and to perform the tests defined in the acceptance criteria. A fundamental aspect they expose is that without the completed UI definition, the team can't know how difficult the work will be, but by using the diagrams and mockups a rough estimate can be done.

After a case study, Carbon et al. [28] find that onsite costumers and developers are often not usability experts; this makes them focus only on functionality when performing the acceptance tests. To improve this situation, they introduce the concept of usability criteria and usability user stories. The onsite customer gets advised by a usability expert in order to define acceptance criteria related to usability, which is called usability criteria. These

usability criteria are integrated into existing and sometimes new user stories which will be called usability user stories. The importance of defining this is to test the usability recommendations when the user stories are implemented.

Although our research builds on the idea of introducing usability functionality, specifically usability mechanisms, into the agile methodologies it differentiates in some aspects from these related works. First of all, we try to build a framework that can be used by teams with no usability experts (as many small-medium companies do not have them). Then we will focus on the incorporation of particular usability recommendations into the agile requirements, which has not been addressed in those previous works. However, it is a complementary line of action which can be used in conjunction with the previous works.

4. OBJECTIVES

The main objective of this research is to build and asses a framework for adding usability mechanisms into agile requirements represented in the form of user stories. This framework is composed by:

- A well-defined modeling language that defines from a high-level perspective, which is the effect of adding usability mechanisms into user stories.
- Instances of the previous modeling language that define how the addition of each specific usability mechanism will affect a user story.
- An open source tool that will automate the process of incorporating usability mechanisms into user stories and simplify its use.

To fulfill this main objective we defined a set of concrete objectives:

- O1: Create a metamodel that defines formally the implications of adding usability mechanisms into a user story.
- O2: Create a well-defined modeling language based on this metamodel.
- O3: Create a set of models based on this language that define the implications of each of the identified usability mechanisms.
- O4: Build a tool that developers can use to automate the process of adding usability mechanisms to the user stories of the software that they develop.
- O5: Asses the use of the tool and therefore the well-defined modeling language in a set of projects.

These concrete objectives were done sequentially. Figure 1 shows an activity diagram that describes the order in which the specific objectives were solved during this research.

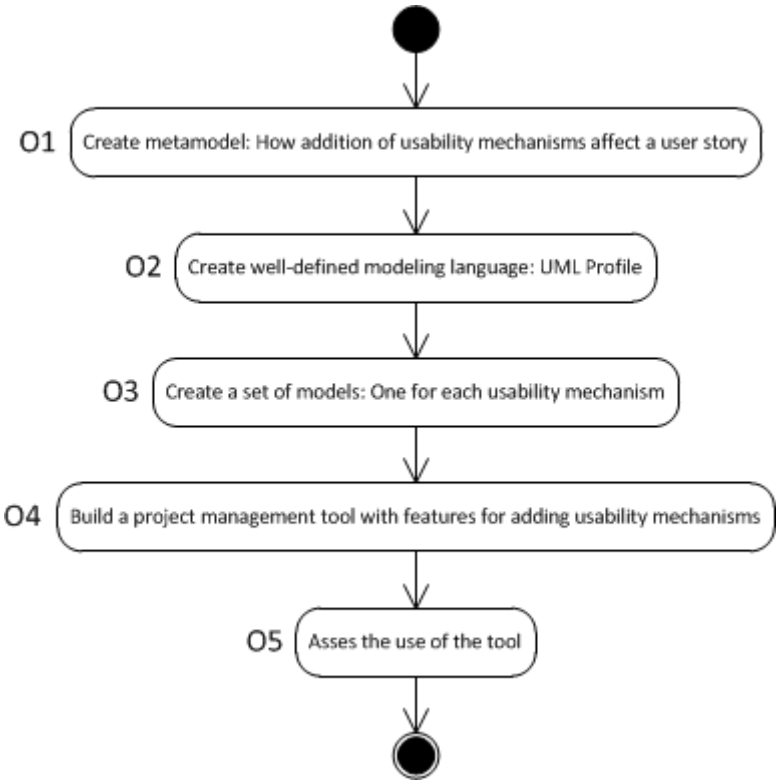

FIGURE 1. ACTIVITY DIAGRAM OF OBJECTIVES

5. METHODOLOGY

A scrum like process was used to carry out the work described in this thesis. The work has been split into two week sprints, with demonstrations and reviews at the end of each sprint. Furthermore, the work has been split into user stories and then into tasks, which have been estimated and prioritized by the stakeholders of the project.

The project was divided in two phases. Phase 1 was centered on reviewing and developing the theoretical part of the framework. Phase 2 was focused on the development and assessment of the open source tool.

On Phase 1 we started by reviewing the previous works related to the incorporation of usability mechanisms into agile user requirements. Based on these previous works we developed the metamodel (O1) using UML modeling tools. We then passed to research about the creation and use of well-defined modeling languages based on UML, specifically UML profile. We converted our metamodel in a UML profile (O2) to define the language and then generated the specific models for each of the usability mechanisms we address (O3).

On Phase 2 we started the development of the open source tool (O4). We took as a starting point an already existing open source scrum management tool named Kunagi. All the features to automatically manage the implications of introducing the usability mechanisms were added to the tool, based on the instances generated on Phase 1. Finally an assessment of the use of this tool was performed (O5).

For the assessment, the tool was given to a set of software engineering master students, so they used it to manage the projects they had to develop. The projects consisted in developing a web application to manage ideas, view them and support them. The students followed a Scrum development process, so the assessment of the tool was documented during the sprint retrospective and demonstration meetings. During these meetings, special attention was given to the way in which the students use the features of the tool that give them recommendations of how to add the usability mechanisms to their user stories. In addition to this, a questionnaire was given to the students in order to receive more information on their experience using the tool.

In total, for fulfilling objectives O1 to O4, we worked during 7 sprints. Table 2 shows the user stories developed during each sprint. O5 was not developed using a scrum like process because the data of the validation had to be treated as soon as it was received and it was done after all the sprints.

Sprint 1 (Phase 1)
US1: Define metamodel
US2: Define “System Status” model
US3: Define “Warning” model
US4: Define “Long Action” model
US5: Define “Go Back” model
Sprint 2 (Phase 1)
US6: Corrections of models due to independent/dependent usability tasks
US7: Define “Text Entry” model
US8: Define “Step by Step” model
US9: Define “Preferences” model
US10: Define “Favorites” model
US11: Define “Help” model
US12: Define “Undo” model
US13: Define “Undo Reset” model
US14: Research about UML Profiles
Sprint 3 (Phase 1 and Phase 2)
US15: Define “Abort Command” model
US16: Define “Abort operation” model
US17: Create UML Profile
US18: Create UML Profile
US19: Study the open source tool and define the user stories for the development phase
US20: Add acceptance criteria functionality
Sprint 4 (Phase 2)
US21: Add Usability Mechanism Entities
Sprint 5 (Phase 2)
US22: Add functionality to automatically add acceptance criteria
US23: Add functionality to automatically add independent usability tasks
US24: Add functionality to automatically add dependent usability tasks
Sprint 6 (Phase 2)
US25: Add functionality to automatically add usability user stories
US26: Add functionality to advise the users when removing usability mechanisms
US27: Create user manual
Sprint 7 (Phase 2)
US28: Add tooltips and Help functionality

US29: Improve UI

TABLE 2. SPRINTS AND USER STORIES OF THE PROJECT

We documented the process while we were working on each of the steps, first continuously documenting the steps in the project management tool Trello [29] and then documenting formally the results after each Phase.

6. PRELIMINARY WORK

In the idea of the combination of UX design and agile methodologies, we take as starting point the work presented by Moreno et al. [30]. In their work, they state that we have three ways in which the incorporation of usability features affects user stories: addition of user stories, addition or modification of tasks and addition or modification of acceptance criteria. Additionally, they develop a table on which they identify the actions to be done when adding each of the identified usability features; this is shown in Table 3. Finally they develop a project management tool on which they implemented the use of their findings, early feedback of the validation of this tool indicates that developers don't require much previous usability knowledge, but there are some issues incorporating the usability discussion into the regular user story creation flow [30].

	New Task	Modify Task	New Acceptance Criteria	Modify Acceptance Criteria	New Usability Story	New User Story
System Status		X	X	X	X	
Warning	X		X	X	X	
Long Action		X	X		X	
Abort command	X	X	X		X	
Abort operation		X	X			
Go Back	X	X	X			
Text entry		X	X			
Step by Step	X		X	X		
Preferences						X
Favorites		X	X		X	
Help		X	X		X	

TABLE 3. MAPPING BETWEEN USABILITY MECHANISMS AND ACTIONS [30]

The work presented by Moreno et al. [30] served as prove of concept of the impact of usability mechanisms into agile requirements. Their research and the tool they developed validated, using an empirical approach, the findings presented in Table 3. Now, we aim to formalize these findings in a well-defined modeling language. These will let us model the impact of the usability mechanisms into the user stories in order to develop a repeatable process in which developers will be able to add usability mechanisms into agile requirements and model its impact. Also, the process and the models created are aimed to be used as base to the development of any project management tool that can be integrated with any kind of agile methodology.

With the implementation of a new project management tool to manage the addition of usability mechanisms in agile methodologies, we aim to solve some of the problems of the tool presented in the work of Moreno et al. [30]. These problems include mainly the easy diffusion on modern servers, thinking on the easiness of use and growth of the tool. Also, it

is an advantage that the tool will be based in formal documentation given by the models created in this research.

7. USABILITY MECHANISMS METAMODEL

As discussed in earlier sections, Moreno et al. [30] give a set of guidelines that describe the effects of adding usability mechanisms to user stories. Based on empirical observations of the use of the software tool developed at [30] we decided to modify the guidelines given in their research in order to give more flexibility to the developers in the management of tasks. The resulting guidelines are the following:

1. Addition of new user stories to represent requirements directly derived from usability. We call these new stories "usability user stories" to distinguish them from traditional user stories, as they represent usability features to be provided by the system.
2. Addition of independent usability tasks. This means that some actions that are strictly derived from usability constraints should be performed in a user story. We call them independent usability tasks, as they don't affect other tasks that might be already added to the user story.
3. Addition of dependent usability tasks. Some actions that are derived from usability constraints should be performed in user story, these dependent usability tasks affect other tasks that might be already added to the user story.
4. Addition or modification of acceptance criteria. These acceptance criteria appear because the user story functionality needs to include some specific actions that modify the operating environment.

The sub-classification of usability tasks into dependent and independent usability tasks aims for two advantages. First, in the theoretical part, distinguish between dependent tasks which will affect other tasks that are already defined and independent tasks which will be totally new to the user story. Then, in the practical part, it allows developers to decide how to approach the addition of a dependent task, either by the addition of a new task or by modifying a previously created task. That way we aim to give more flexibility to the developers.

Our goal is to define a formal way to work with these guidelines. We decided to do this by developing a UML metamodel that comprises them. The metamodel describes the addition of any usability mechanism by showing its relations with the other entities of the agile methodology domain. This resulting metamodel is intended for defining an instance for each of the addressed usability mechanisms. The metamodel is shown in Figure 2.

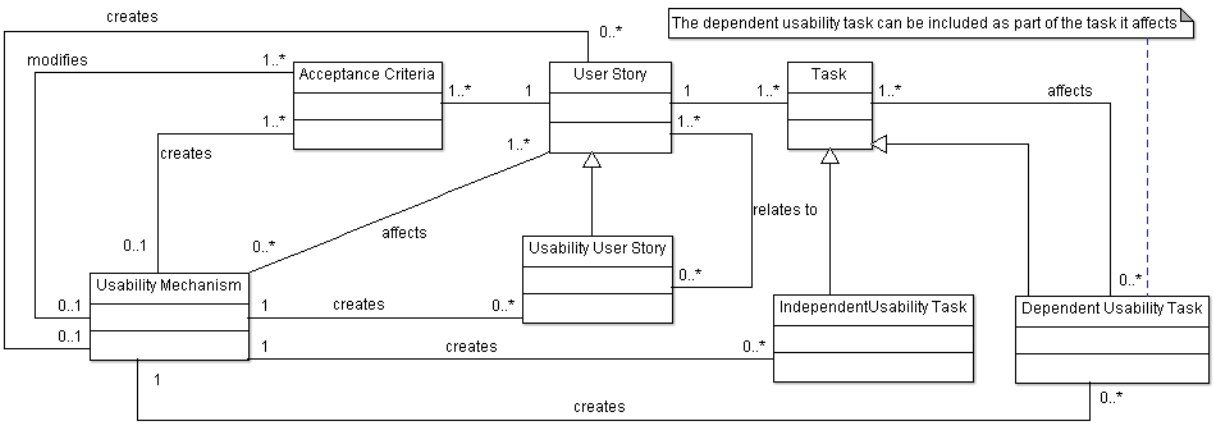

FIGURE 2. METAMODEL

The creation of this metamodel fulfills O1 as set in Figure 1.

8. USABILITY MECHANISMS PROFILE

Taking in consideration that we want to develop a set of instances of the metamodel inside the specific domain of adding Usability Mechanisms into agile requirements, we decided to create an UML profile that defines a well-defined modeling language. A well-defined language is a language with syntax and semantics, and has an advantage that is also suitable for interpretation by a computer [31]. The language we create is aimed to improve the modeling process and the readability of the resulting models.

An UML profile is a set of extension mechanisms (stereotypes, tagged values and constraints) that allow customized extensions of the UML for a particular domain [31]. In our case these extension mechanisms will define the domain of the impact of Usability Mechanisms inside the agile process.

One of the advantages of creating this UML profile is that UML profiles always describe a well-formed model for a domain, and ensure that the specific models that use it will always comply with the syntactic or semantic constraints defined by it [31]. So, this profile will ensure that all the models and instances that use it comply with the domain. Also it will give the possibility create other models inside the domain of the profile when more usability mechanisms are identified.

Another benefit of defining the domain in a UML profile is that all the resulting models could be used in commercial UML tools for drawing diagrams, code generation, reverse engineering, among others [31].

Fuentes-Fernandez and Vallecillo-Moreno [31] define the process of generating an UML profile as a set of steps. We took these steps as the methodology to build the UML profile.

The first step is to define the elements and relationships that comprise the system; they also recommend performing this step by defining a metamodel as if it was not intended for building an UML profile [31]. By looking at the metamodel in Figure 2, we observe that it is a set of entities and relationships that define a specific language in the domain of adding Usability Mechanisms into agile requirements; therefore it fulfills this first step.

The second step is to add a stereotype for each of the entities we have in the metamodel and step three is to identify the elements in the UML metamodel that each of our stereotypes is extending [31]. These two steps lead us into defining the UML package shown in Figure 3. The entities defined in the general metamodel extend the UML metaclass that defines a *Class*, except for *Usability User Story* which is a specialization of a

User Story and *Independent Usability Task* and *Dependent Usability Task* which are specializations of a *Task*. The relationships defined in the metamodel extend the UML metaclass that defines an *Association*.

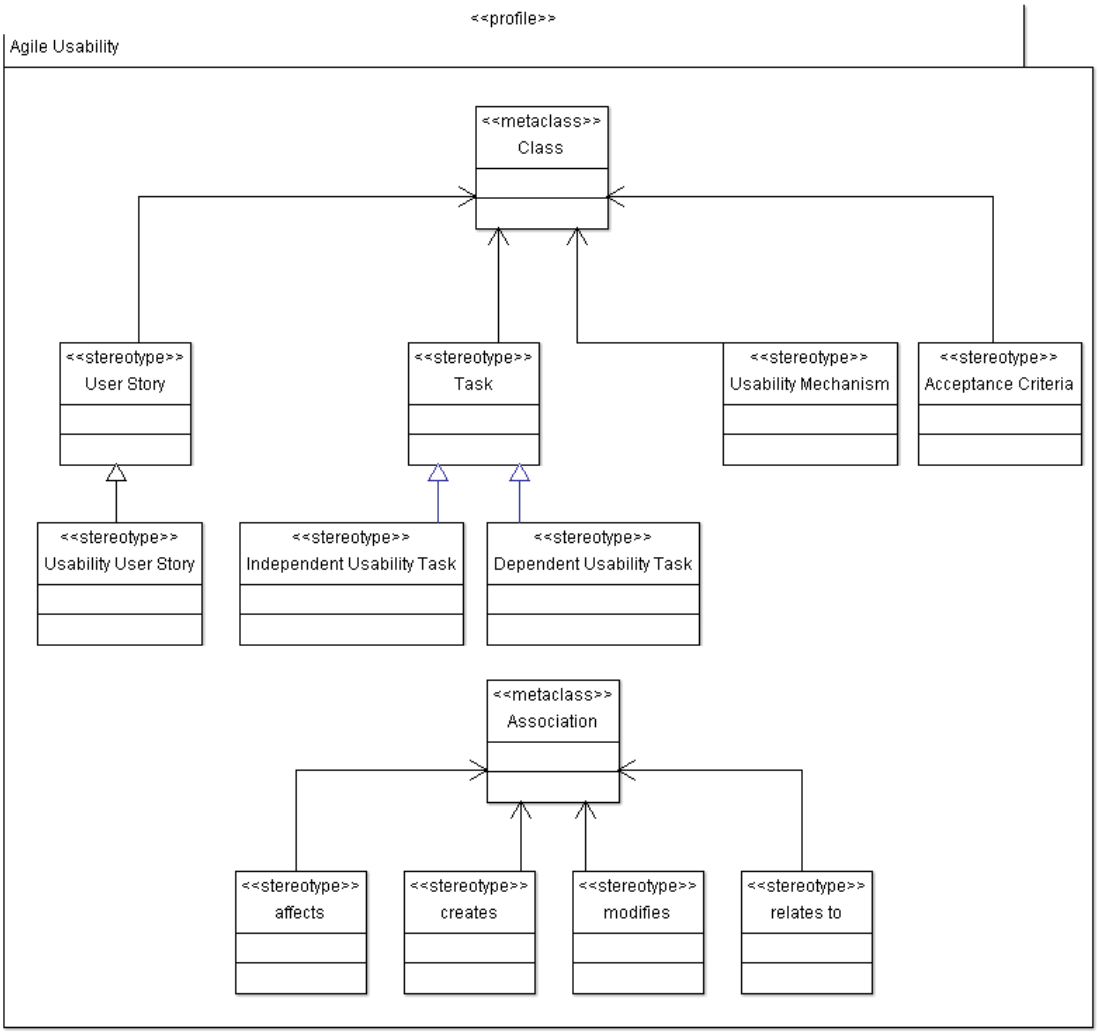

FIGURE 3. UML PROFILE DESCRIBED AS UML PACKAGE

Step number four of the UML profile creation is to use the attributes defined in the metamodel to identify the tagged values of the profile [31]. In the case of our metamodel there are no attributes, so the UML profile will not have tagged values.

The last step is to define the constraints of the UML profile from the domain restrictions [28]. In the case of the general metamodel we defined the following restrictions:

1. Association multiplicities.
2. Allowed relations of each stereotype.
3. Allowed participant stereotypes in each relation.

These constraints where modeled in Object Constraint Language (OCL). Restrictions 1 and 2 were modeled by extending the UML Class element. For example the allowed multiplicities and relations of a class stereotyped as *Usability Mechanism* were modeled as follows:

```
context UML::InfrastructureLibrary::Core::Constructs::Class

	inv :

	self.isStereotyped("Usability Mechanism")

	implies

		self.connection->select(isStereotyped("affects"))->size > 0

		and self.connection->select(isStereotyped("creates"))->size >= 0

		and self.connection->select(isStereotyped("modifies"))->size >= 0

		and self.connection->reject(isStereotyped("affects") or
	isStereotyped("creates") or isStereotyped("modifies"))->isEmpty
```

It defines the allowed values in the association ends in the first 3 lines of the *implies* section. The last line of the *implies* section defines the only allowed relations that can be added to a class stereotyped as *Usability Mechanism* which are *affects*, *creates* and *modifies*, this can be further verified with the metamodel.

Restriction 3 was modeled by extending the UML Association element. For example the allowed stereotypes that can participate into an *affects* relationship are modeled as follows:

```
context UML::InfrastructureLibrary::Core::Constructs::Association

		inv :self.isStereotyped("affects")

		implies

			self.connection->reject(c1, c2 |
	(c1.participant.isStereotyped("Usability Mechanism") and

			c2.participant.isStereotyped("User Story")) or
	(c1.participant.isStereotyped("Task") and

			c2.participant.isStereotyped("Dependent Usability
	Task")))->isEmpty
```

This defines that the only allowed *affects* relationships are either between a class stereotyped as *Usability Mechanism* and a class stereotyped as *User Story* or between a

class stereotyped as *Task* and a class stereotyped as *Dependent Usability Task*, this can be further verified with the metamodel.

The full OCL constraint model for the UML profile is shown in Appendix A.

The whole definition of the UML profile with its stereotypes and OCL restrictions fulfills O2, as set in Figure 1.

9. USABILITY MECHANISM MODELS

9.1 DEFINITION OF THE MODELS

As mentioned before Juristo et al. [13], give a set of usability mechanisms that might be added to systems, and therefore affect the requirement process and products The work presented by Moreno et al. [30] summarizes the implications of adding most of this Usability Mechanisms to user stories as we showed in Table 3 on section 5.

We took this table (Table 3), verifying it with the tool implemented by Moreno et al. [30] and used the information given for each usability mechanism in the work of Juristo et al. [13] to create the usability mechanism models. These models have the advantage of representing the process of adding each usability mechanism to a user story in a well-defined modeling language, thus defining a process that can be repeated every time that this Usability Mechanism is used. The well-defined modeling language is given by the UML profile created in the previous section.

Let's look at the example of the Warning Usability Mechanism. The full definition of the Warning Usability Mechanism given by Juristo et al. [13] is shown in Table 4. Table 3 shows that, according to the findings of Moreno et al. [30], the addition of the Warning Usability Mechanism implies adding a new task, adding new acceptance criteria, modifying acceptance criteria and adding a new usability user story. Table 4 gives information that can help us find why these changes are added, for example, the *HCI Recommendations* section of the table gives information of how the warning should be shown and what options the user should have when looking at the warning. This will be used to add comments in the model so the developers that use it can know exactly what tasks and acceptance criteria to add and modify when adding the Warning Usability Mechanism to a user story and therefore follow a well-defined process.

Figure 4 shows the resulting model for the Warning Usability Mechanism. This model can be read as follows, the Warning Usability Mechanism affects the User Story by creating a d*ependent usability task* that affects another task. The relation of these tasks is due to the fact of inserting the Warning inside the User Story, thus it specifies what will be the warning about and that it appears at the proper moment of the system activities. Furthermore, the Usability Mechanism modifies the acceptance criteria of the User Story by having to check that the Warning emerges at the right moment; that is, when the event to warn about happens. Other acceptance criteria are created in order to ensure that the

warning is obtrusive, and that the actions performed after the user interacts with the warning work properly.

<table>
<tr><td colspan="2">IDENTIFICATION</td></tr>
<tr><td colspan="2">Name: Warning</td></tr>
<tr><td colspan="2">Family: Feedback</td></tr>
<tr><td colspan="2">Alias: Warning
Think Twice</td></tr>
<tr><td colspan="2">PROBLEM</td></tr>
<tr><td colspan="2">Which information needs to be elicited and specified in order to ask for user confirmation in case the action requested has irreversible consequences.</td></tr>
<tr><td colspan="2">CONTEXT</td></tr>
<tr><td colspan="2">When an action that has serious consequences has been required by the user.</td></tr>
<tr><td colspan="2">SOLUTION</td></tr>
<tr><td colspan="2">Usability Mechanism Elicitation Guide</td></tr>
<tr><td>HCI Recommendation</td><td>Issues to be discussed with stakeholders</td></tr>
<tr><td>1. For each action that a user may take, having regard for: the reversibility of the action, the proportion of reversible actions that the system supports, the frequency with which the action is taken, the degree of damage that may be caused, and the immediacy of feedback, consider whether a warning, confirmation or authorization may be appropriate.</td><td>1.1. Discuss with the user all task to be performed and their consequences (consider the frequency of such actions and its damage) and which do require confirmation (be careful not to overload the user with warnings)</td></tr>
</table>

2. Users may not understand the consequences of their actions neither other options to perform. So, the warning should contain the following elements: – A summary of the problem and the condition that has triggered the warning. – A question asking the users if continuing with the action or take on other actions. Two main choices for the user, an affirmative and a choice to abort. – It might also include a more detailed description of the situation to help the user make the appropriate decision. The choices should state including a verb that refers to the action wanted. – In some cases there may be more than two choices. Increasing the number of choices may be acceptable in some cases but strive to minimize the number of choices.	*2.1. Which information will be provided for each of the tasks to confirm? Remember to provide the consequences of each action and the alternatives to the user.*
Usability Mechanism Specification Guide	
The following information will need to be instantiated in the requirements document: Tasks U, V, Z, will require warning. For them the information to be shown will be R, S, T, respectively	
RELATED PATTERNS	
Abort Operation: One of the alternatives of the warning message should be to Cancel the action	

TABLE 4. DEFINITION OF WARNING USABILITY MECHANISM [13]

The implementation of the Warning window is itself another User Story, specifically a Usability User Story which can be reused when adding the Warning Usability Mechanism to other user stories of the system. The model specifies that this Usability User Story is created by the addition of the Warning Usability Mechanism and it is related to the affected User Story.

The Usability User Story has tasks and acceptance criteria of its own. All of these tasks have to do with the implementation of the User Interface part of the Warning and the acceptance criteria ensure its proper functionality, the specific details of each are described in the model.

The full set of specific Usability Mechanism Models is shown in Appendix B. These models fulfill O3, as set in Figure 1.

9.2 Advantages of Using the UML Profile

Another important aspect to see from these specific models is the use of the UML profile. Figure 5 shows the Warning model without the use of the profile, in order to compare it with Figure 4, where we see it with the profile. We can see that the readability of Figure 4 (with the UML profile) is better than the one of Figure 5 (without the profile), this readability is further perceived when defining instances as it is shown in the next section.

Each stereotype gives a meaning to every entity in a well-defined modeling language that is defined by the profile and its constraints. In figure 4, each profile let us know exactly the meaning of each class and each relation of the model inside the agile development process. In figure 5 we can only guess it by the name of the classes and relations.

Finally, the profile puts restrictions that could also be used to create other models for other Usability Mechanisms that might not be addressed in this research and to create instances of the models for more specific purposes like modeling the user stories of the application. These restrictions make all the models that use the profile to follow the rules described by it, so they make sense inside the scope defined by the language; that is inside the restrictions of the relationships between user stories, tasks and acceptance criteria given by the agile methodologies and extended by the definitions of this research.

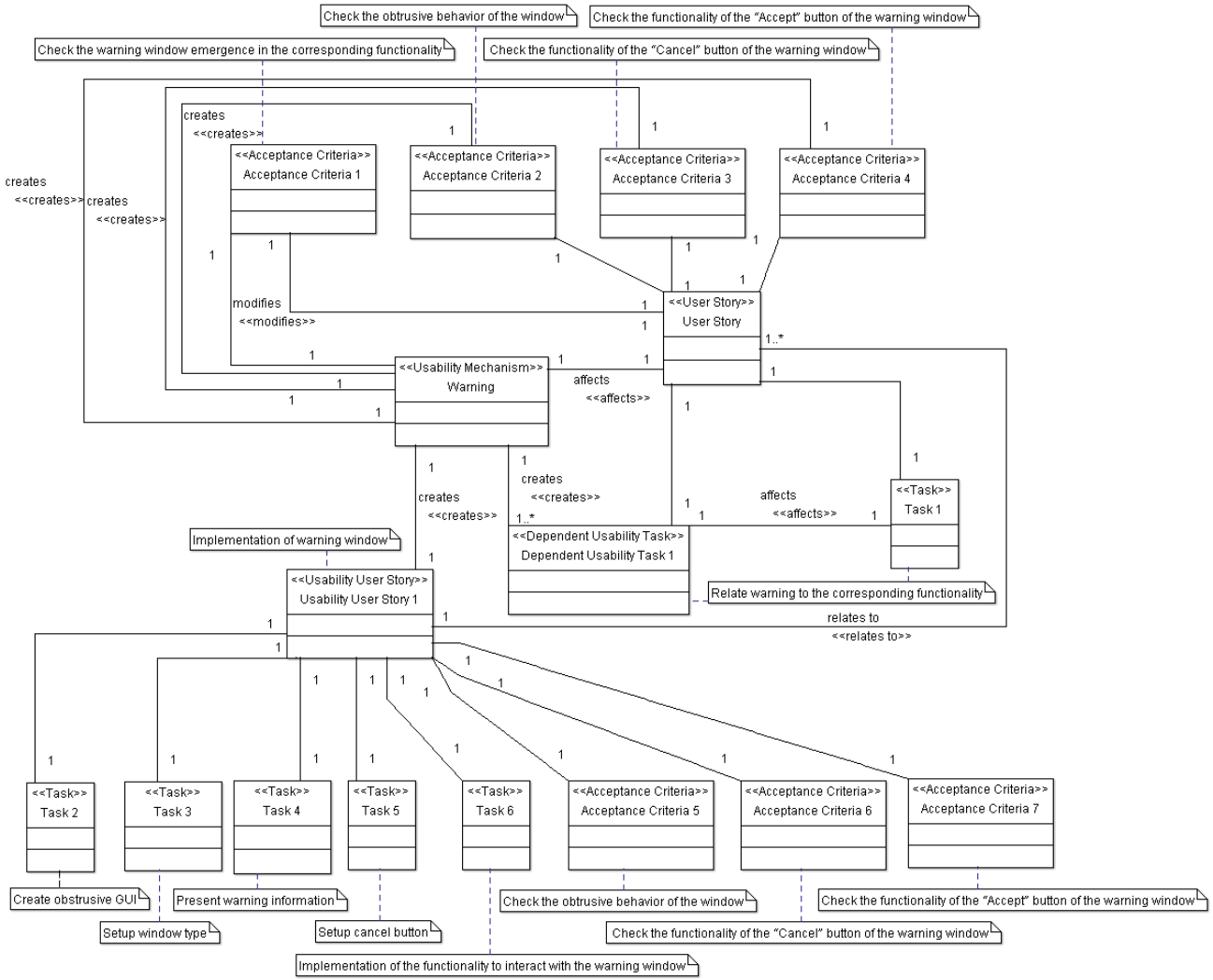

FIGURE 4. WARNING MODEL

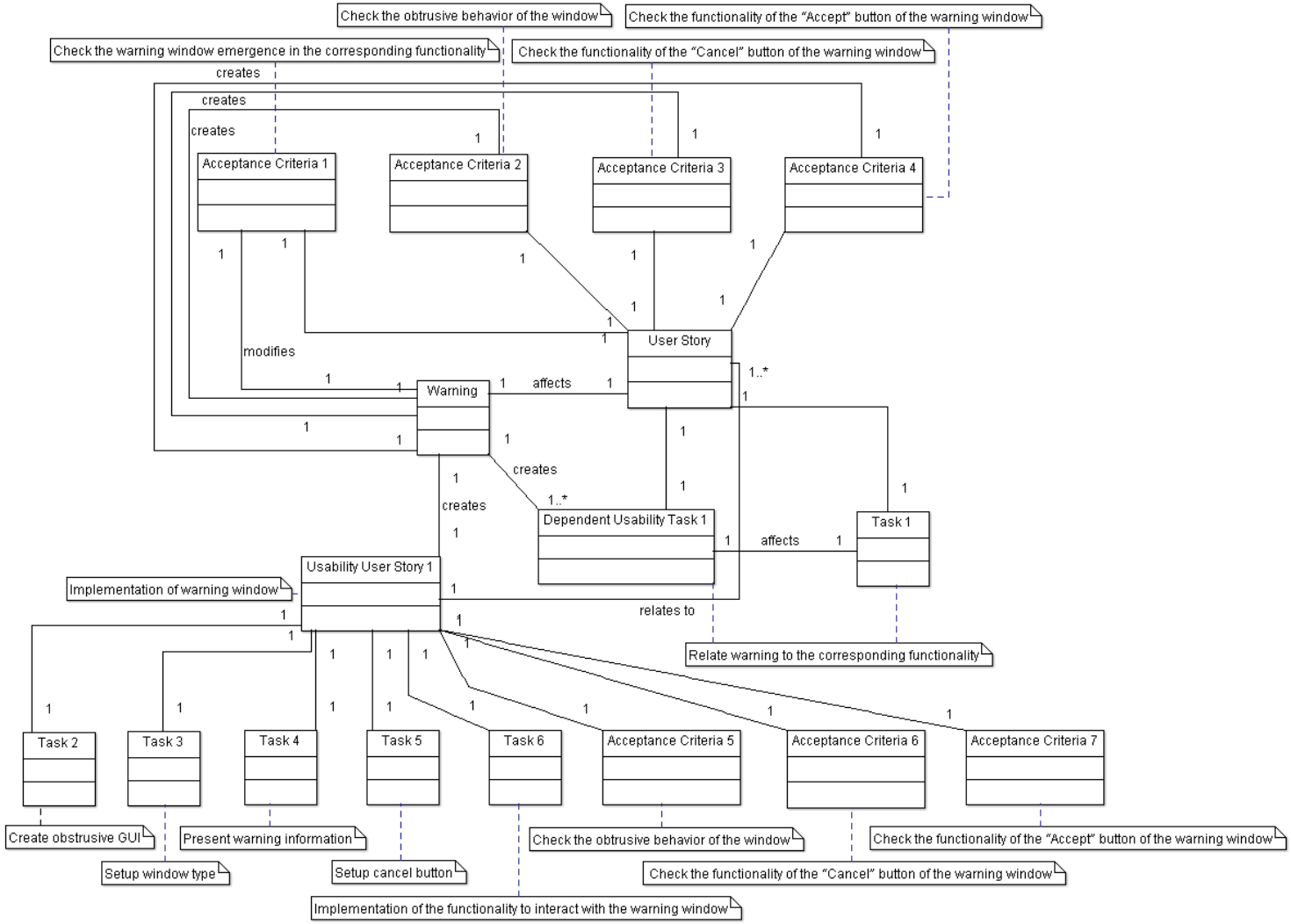

FIGURE 5. WARNING MODEL WITHOUT UML PROFILE

9.3 DEFINITION OF INSTANCES

When a developer will use one of these models to add usability mechanisms to his user stories he will be able to define an instance of the model that is specific to his application scope, that is, the user story he wants to enhance. This is, as stated before, an advantage of using the profile; it gives the possibility to create more specific models within the domain.

A developer will start with a user story like the following: *As a user I want to be able to delete files*. It will have certain tasks and acceptance criteria, for example a model for this user story could be the one shown in Figure 6.

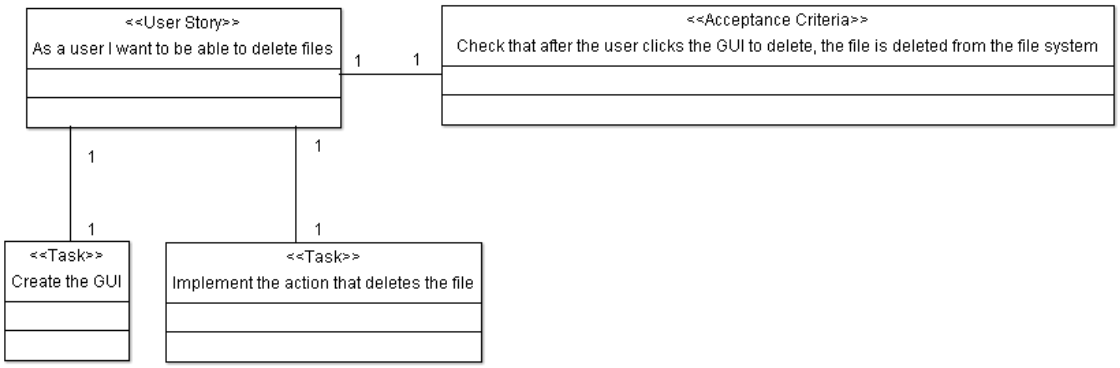

FIGURE 6. INSTANCE OF USER STORY

Then we can suppose that the developer will want to add the Warning usability mechanism to this user story, in order to alert the user when he is deleting the file. The user story on Figure 6, will be the user story that is affected by the Warning usability mechanism on the Warning model, shown on Figure 4. Then the task *Create GUI* will be the one affected by the dependent usability task, because when the user clicks the GUI that will call the action of the deletion of the file the warning must be shown first. The acceptance criteria *Check that after the user clicks the GUI to delete, the file is deleted from the file system* will be modified by the Warning usability mechanism, because instead of deleting the file at that moment, the Warning message should appear, and the check of the deletion of the file will be done when the user clicks the accept button. Finally, the Usability User Story of the implementation of the warning message will be added, as is, to the instance model and related to the user story, because it is supposed that it is a user story that will be reused whenever a warning message will be needed throughout the application. Figure 7 shows the resulting instantiated model, the classes added or modified because of the usability mechanism are marked in grey. Note that the whole instance of the usability user story of the implementation of the warning window is the same one that

would appear in any user story that uses a warning, so it means only one time of extra work when implementing the warning functionality.

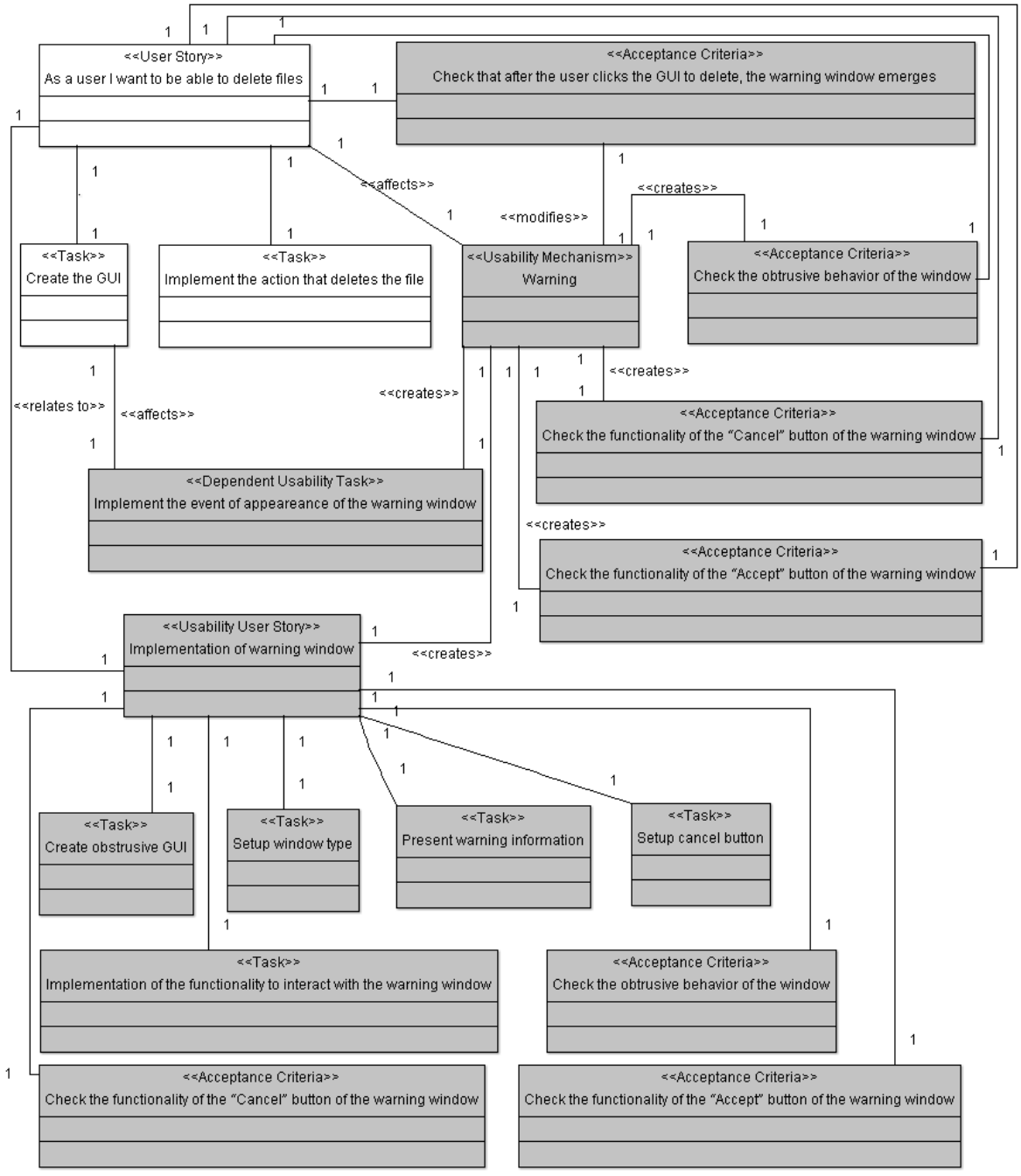

FIGURE 7. INSTANCE OF USER STORY WITH WARNING USABILITY MECHANISM

10. TOOL SUPPORT

10.1 TOOL AIM

The next step in our research was to develop a scrum project management tool that comprises the process described in the previous sections (O4). We decided to extend an open source scrum management tool, this way we focus only in the features that are to be added in order to insert usability mechanisms to user stories. The summarized set of features to be added to the tool is the following:

- Management of usability mechanisms added to each user story
- List of usability recommendations for the tasks of a user story by usability mechanism, and addition of dependent or independent usability tasks if needed by the user
- List of usability recommendations for the acceptance criteria of a user story by usability mechanism and addition or modification of acceptance criteria if needed by the user
- Automatically addition of usability user stories by usability mechanism
- Help functionality for these features

The final contribution of this tool is to help the developers in the process of adding usability mechanisms to the user stories that they implement in the application without needing to have previous knowledge of the process and modeling languages that define it. Therefore we aim to add the least extra work as possible to the developers using it.

10.2 TOOL SELECTION PROCESS

For the selection of the open source tool to extend we took in consideration some important characteristics. The first was that the tool was implemented in a well-known language, in this case we chose Java. This was important in order to ensure easy diffusion of the tool in the topic of easy deployment of the application and servers needed and in the topic of growth of the tool. After this first filter, three tools were considered.

The first tool reviewed was Agilefant [32]. This tool has all the functionality of the scrum development process except for the management of the acceptance criteria, which is one disadvantage. We looked at the code and it was not so trivial to develop this missing functionality due to the complexity of the implementation of the tool and its lack of documentation. Another disadvantage was that, in our perspective, the tool was not so user friendly, which in the final state of the tool is especially important for this research.

The second tool that we considered was IceScrum [33]. This tool had all the functionality of the scrum development process, even the management of acceptance criteria. The main disadvantage was usability. In this tool, there is a lot of bureaucracy in order to work with the user stories and to break them into tasks. Also there is a pro version of the tool, and we preferred a tool that is totally open source and that is sure it will remain that way.

The final tool considered and the one that was chosen was Kunagi [34]. This tool also has all the functionality needed for the scrum development process even acceptance criteria management. The problem is that the acceptance criteria in Kunagi are managed in a different way than what was expected by the agile principles used in the scope of this research. Acceptance criteria were added as just one text description inside the user story, while we wanted to give the user the possibility to have more than one acceptance criteria per user story if needed. Even thought, we proceeded to review the code and the addition of this functionality was straight forward. Also the original tool, follows a lot of the usability recommendations used in this research, therefore it seemed like a good enough tool to choose. The summary of the tool selection analysis can be seen in Table 5.

Tool	Starting Functionality	Documentation	Usability	Complexity in modifications
Agilefant	Lack of Acceptance Criteria functionality	No	Medium	High
IceScum	Full functionality	No	Low	High
Kunagi	Lack of Acceptance Criteria functionality	No	Acceptable	Low

TABLE 5. SUMMARY OF TOOL SELECTION ANALYSIS

10.3 Description Of U-Kunagi

This section explains the use of the tool after the addition of all the new features. The final result of the modification of Kunagi can be found in this link: http://oeixtatil-13.eui.upm.es:8080/kunagi/. We call the final tool U-Kunagi.

The first feature that we needed to add to the tool was the one of management of acceptance criteria. This because of the reasons explained in the past section. The result of this modification is shown in Figure 8.

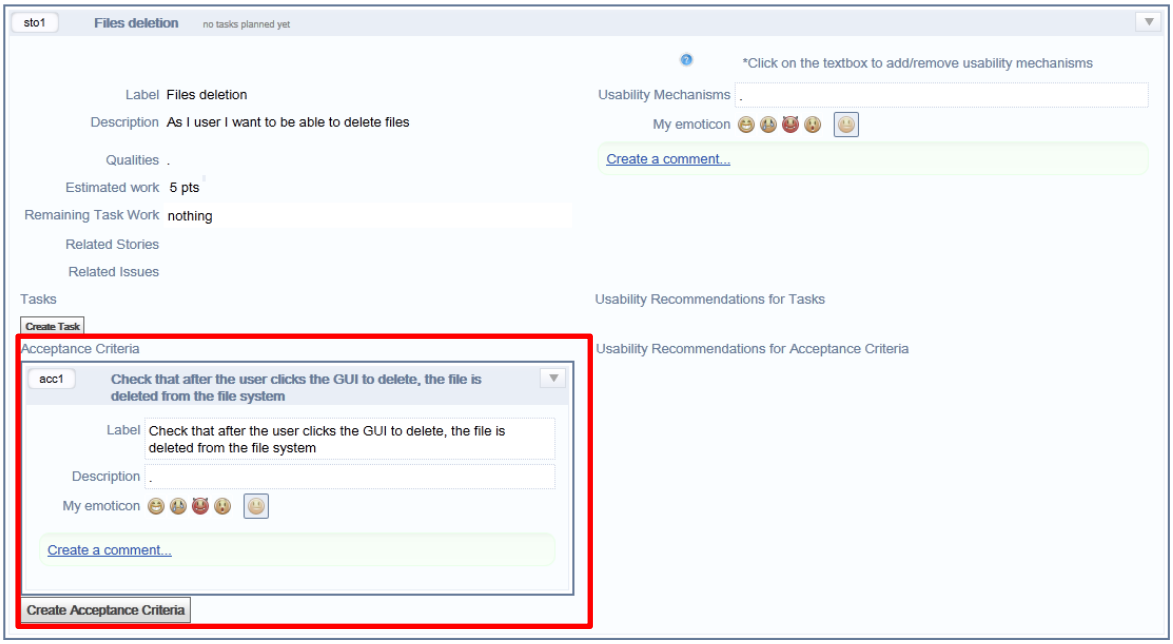

FIGURE 8. ACCEPTANCE CRITERIA MANAGEMENT

After the addition of this feature, the features that are specifically related to the process described on this research were added. So, following the example shown in section 8.3, the user story of the file deletion, with its tasks and acceptance criteria, shown in Figure 6, is instanced in the tool as we can see in figure 9. We have the user story *As a user I want to be able to delete files*, with its two tasks *Create the GUI* and *Implement the action to delete the file* and its acceptance criterion *Check that after the user clicks the GUI to delete, the file is deleted from the file system*.

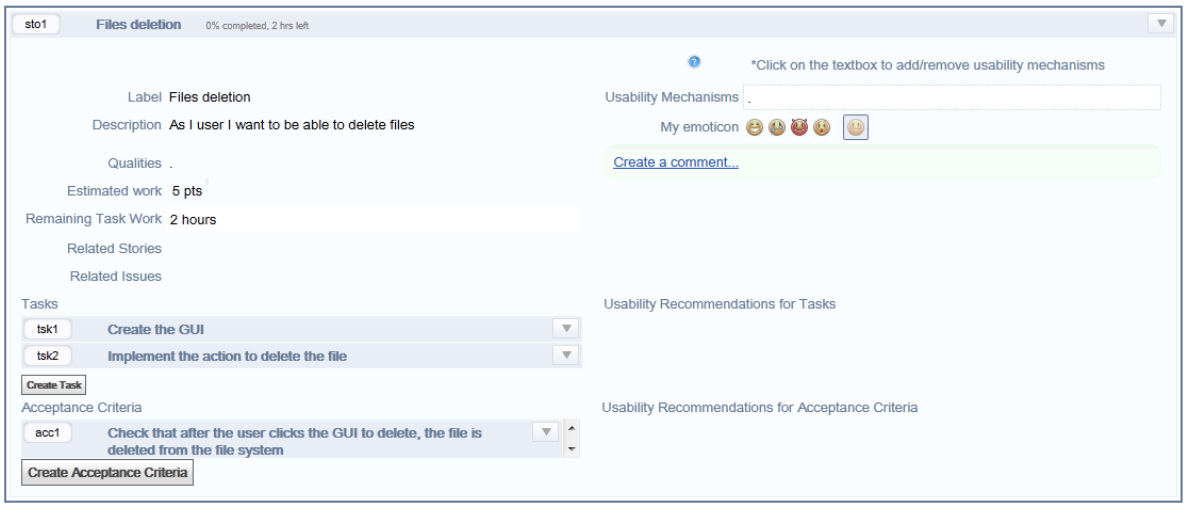

FIGURE 9. INSTANCE OF A USER STORY IN KUNAGI

Then, as we did before on section 8.3, we can assume that the developer will want to add the Warning usability mechanism in order to warn the user about the deletion. The

developer can choose from the usability mechanisms list the one of the Warning. Automatically, the tool will show the usability recommendations for any addition or modification of tasks and the usability recommendations for any addition or modification of acceptance criteria. Furthermore, it will find (or create if is not already added) the usability user story of the implementation of the warning window, and relate it to the user story. All this is shown in Figure 10.

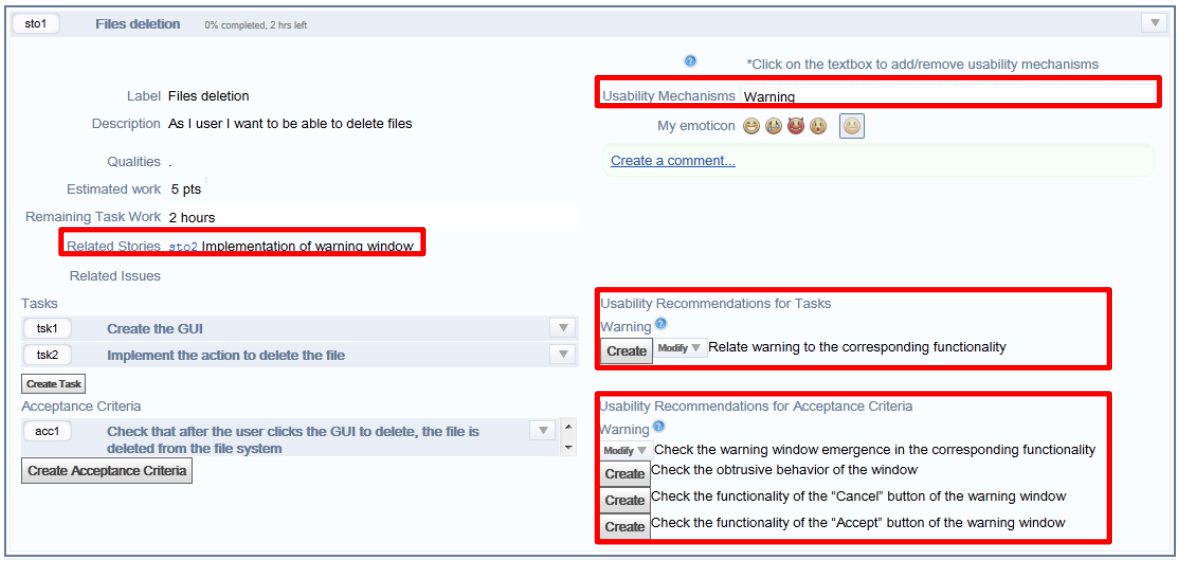

FIGURE 10. OVERVIEW OF USABILITY MECHANISM ADDITION

Each usability recommendation has a button to add it to the user story, this addition can be either a creation or a modification of a task or an acceptance criteria. The developer will add the usability recommendations with these buttons to end up with the instance of the user story as it is affected by the Warning usability mechanism. In this example, we added the dependent usability task with the *Create* button and modified it in order to have a name that means something for the user story, thus having the task *Implement the event of appearance of the warning window*. Then we modified the acceptance criterion with the button *Modify* and modified it to say *Check that after the user clicks the GUI to delete, the warning window emerges*. Finally we added the other acceptance criteria that check the functionality of the warning window in the scope of the user story by using the *Create* buttons in the Usability recommendations section of the acceptance criteria. Figure 11 shows the final instance of the user story as shown in the tool; this user story is the same that is modeled in Figure 7.

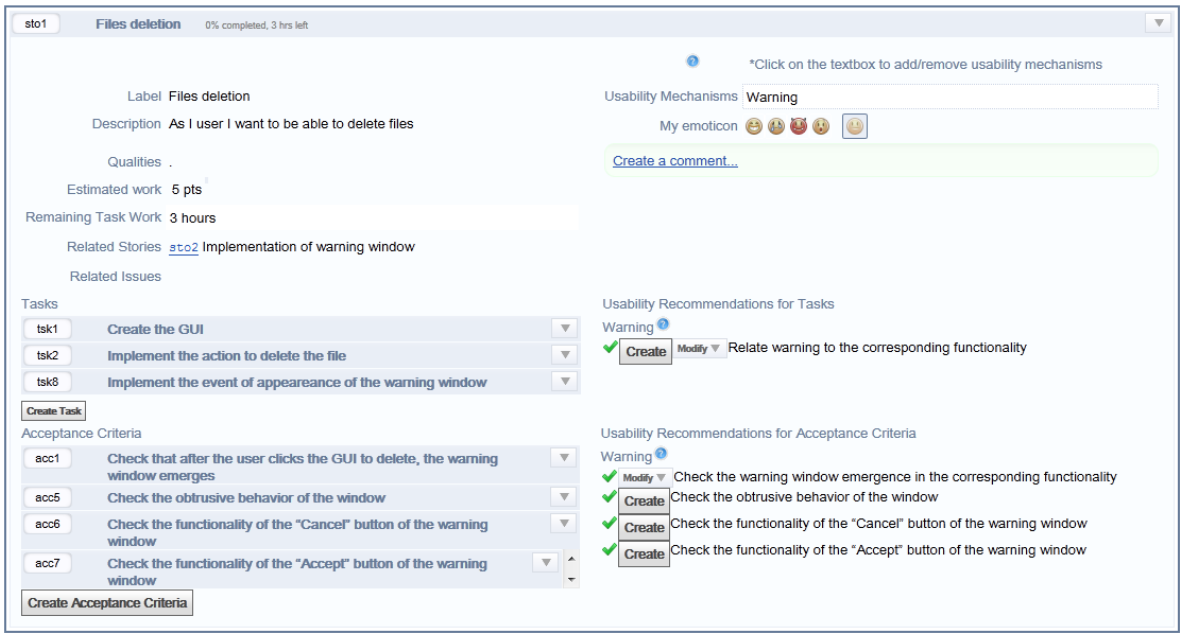

FIGURE 11. FINAL INSTANCE OF THE USER STORY IN KUNAGI

Finally, using the link found in the Related Stories section, the user story of the implementation of the warning window can be viewed. This user story, as appears on the tool is shown in Figure 12. This user story should be reused everywhere in the project were a warning window is needed, thus, it must be implemented only once. Then when another user story needs the waning usability mechanism, this same usability user story will be related to it, it doesn't matter on which iteration it was implemented.

FIGURE 12. USABILITY USER STORY: IMPLEMENTATION OF WARNING WINDOW

10.4 VALIDATION

The tool and the process of using usability mechanisms into agile requirements were applied by the UPM Master on Software Project Management students for developing their Master Thesis. The students developed a web site to manage ideas following an agile approximation, in particular Scrum. In short, the application developed should allow users to upload their ideas and other people view them, support them, comment them, etc. Table 6 shows the backlog of one of the projects [35] in order to demonstrate the basic functionality and size of the applications used for the validation.

Sprint 1
US1: Registration of user
US2: Log-in (Front-end)
US3: Log-in (Back-end)
US4: List my ideas
US5: Create new idea
US6: Attach multimedia to my idea
US7: Delete idea
US8: Modify ideas
US9: Manage profile
US10: Visualize idea
US11: List ideas
US12: List categories (Back-end)
US13: Create categories (Back-end)
US12: Modify categories (Back-end)
US13: Delete categories (Back-end)
US14: RSS Subscription
Sprint 2
US15: Vote for ideas
US16: Tag cloud for categories
US17: Delete my account
US18: Forgot password
US19: Donate money to ideas
US20: Search for ideas
US21: Comment ideas
US22: Attach multimedia to idea
US23: Contact author

US24: Suggest similar ideas
US25: See donations
US26: “Like” for social networks
US27: Graph to show visualizations per day
US28: Upload images
Sprint 3
US28: Encrypted passwords
US29: Ask to change password at least once a year
US30: Block account after 3 tries of entering without success
US31: Policy of privacy
US32: Terms of use
US33: Contact administrator
US34: “About us” page
US35: List users (Back-end)
US36: Create users (Back-end)
US37: Modify users (Back-end)
US38: Delete users (Back-end)
US39: List ideas (Back-end)
US40: Create ideas (Back-end)
US41: Modify ideas (Back-end)
US42: Delete ideas (Back-end)
US43: Email
US44: FAQ
US45: Help video
US46: Help

TABLE 6. PRODUCT BACKLOG OF ONE OF THE PROJECTS USED FOR VALIDATION [35]

Figure 13 shows a user story of one of the projects which uses a Warning usability mechanism. Figure 14 shows a user story of another project which uses the Go Back, System Status and Text Entry usability mechanisms.

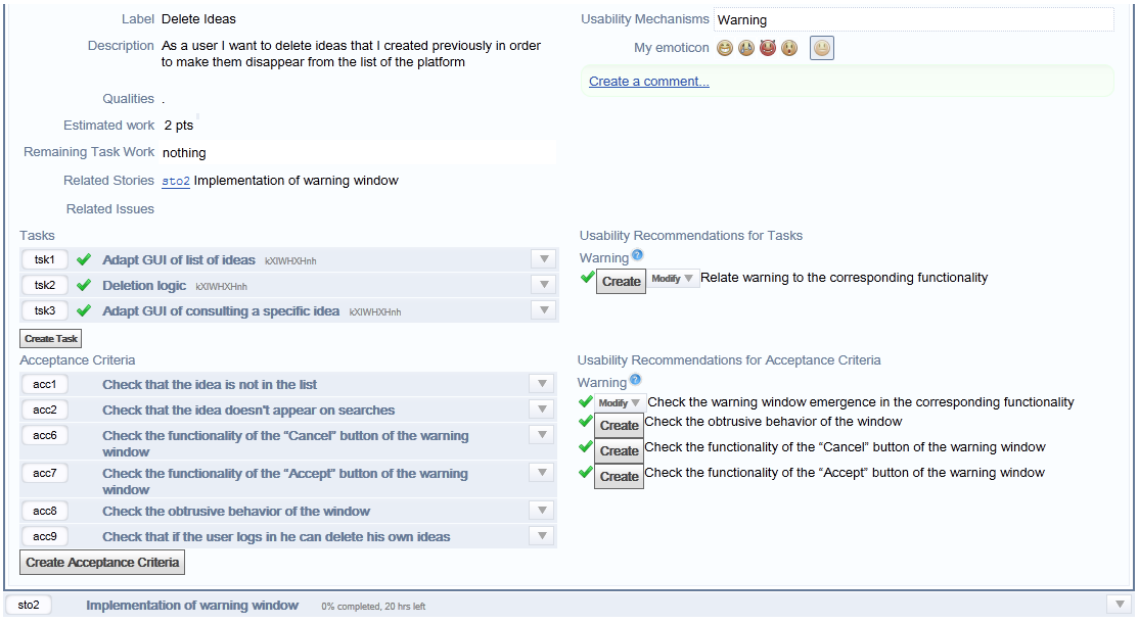

FIGURE 13. USER STORY WITH WARNING USABILITY MECHANISM

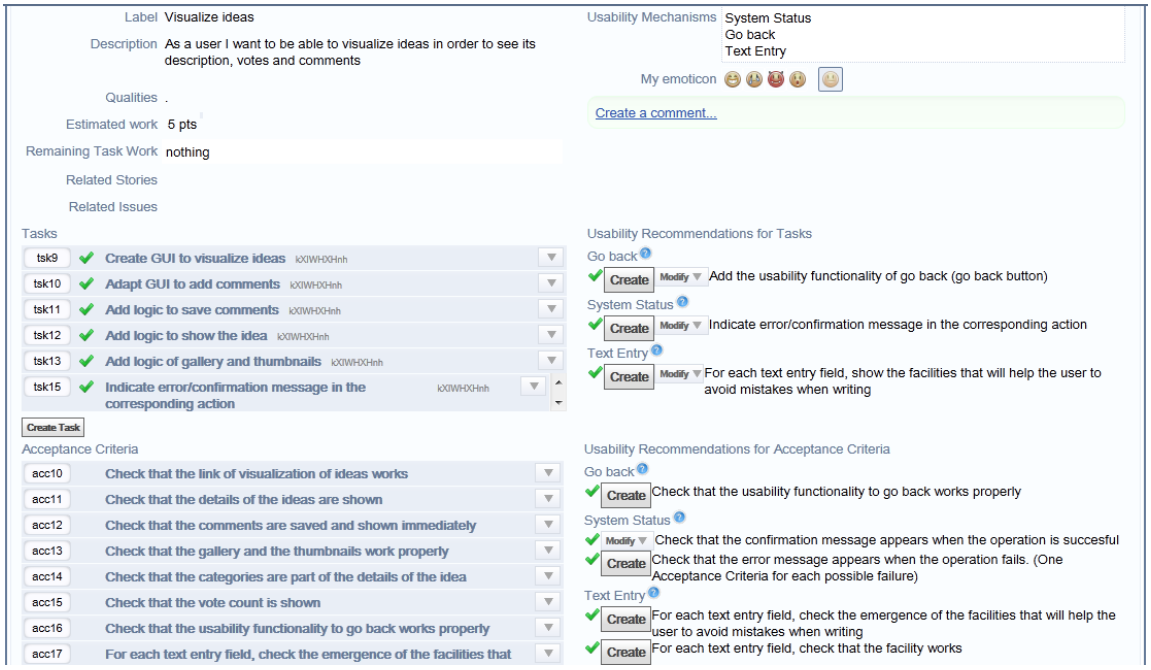

FIGURE 14. USER STORY WITH GO BACK, SYSTEM STATUS AND TEXT ENTRY USABILITY MECHANISMS

The students used the Scrum methodology, so the information about their experience with the tool was carried out during the Scrum Retrospective meetings. The information that we gathered is related to the addition of usability mechanisms, their advantages and disadvantages and about the tool. Also at the end of their projects, a questionnaire (shown in Appendix C) was given to them in order to receive more punctual information about the process.

The students used the usability mechanisms regularly, in more than half of the user stories of their projects. This means that the addition of usability mechanisms is relevant in most of the features of the projects they were working on. Though, they used just a small set of the usability mechanisms provided, mostly the *Help*, *Go Back*, *Warning* and *Text Entry*, probably due to the type of projects and the fact that they were web based.

A good aspect of using the tool and in general the framework is that it didn't demand any important amount of extra effort to the students. They even reported perceiving to save some effort on later iterations, where if not for the usability mechanisms they would have had to change their user stories.

Among the advantages of using the framework described in this research the students pointed out that the tool, the usability mechanisms and their recommendations helped them in order to add aspects that otherwise they would have not though about until later in the project. Also, they stated that the tool helped them even to add aspects that otherwise they would have not thought about, therefore improving the usability of their final project. The students also mentioned that the use of the tool and the methodology will help them to think about usability mechanisms during following projects; even if they don't have a tool that supports them.

On the other side, applying the usability mechanisms in the agile process changes the usual way of working of the students. Due to this, there was a lack of custom of the students in determining the application of usability features needed in their developments. Occasionally, this caused a slight change in the students regarding the addition of usability mechanisms in early iterations. Also, the students said that they were lacking some important usability issues that are more related to the web technologies, like searches, ordering of results, pagination, etc. This last concern was taken in consideration when building the models and the profile, which allow extending the usability mechanisms used simply by creating new models that use the UML profile.

Overall, we observed that the tool helped them to think about usability during the development of the project. They didn't seem to need more usability experience in order to add usability mechanisms into agile requirements, but they seem to need more experience with the approach of thinking about usability early in the project.

11. CONCLUSIONS

As a result of the work presented, we proved and modeled the impact of adding usability mechanisms into agile requirements. These models were developed inside a well-defined modeling language which is given by an UML profile. They serve various purposes, to formally define the impact of usability mechanisms to user stories, to be used to instantiate specific usability mechanisms and user stories that are impacted by them and, due to the use of UML, the models could be used by automatic tools.

Also we created a framework for helping developers to integrate the findings of this research into the agile development process. The framework includes the models, the profile and an open source tool to manage the integration. The tool proved to be useful for its main objective that was helping practitioners integrating the usability mechanisms into the user story definitions. It also helped the developers in order to think about usability aspects that otherwise they have not thought about or they would have thought later in the project when it would have had more impact on the developers' effort [14].

These results draw UX design and agile methodologies closer together, and we can see that thinking about including usability mechanisms into agile requirements can help overcome some of the challenges of this integration. One of the most important contributions of the framework presented in this research is that practitioners don't need to be usability experts in order to add and think about usability features for the software they develop. The tool gives them a set of quick and commonly used usability mechanisms that are aimed to save them time and effort.

Another important contribution is that all the previous empirical research about the impact of the addition of usability mechanisms into agile requirements is now documented into UML models and further empirical research of other usability mechanisms can be documented in the same way just by applying the profile. The use of UML also lets the models to be used to create different kinds of automatic tools to improve the developers experience when using the framework.

Even though, the tool and the process require further improvements which can define further work related to this research. For example, a deeper validation of the process using developers who have different degrees of expertise in agile methodologies is needed. This might help us to answer some of the concerns of the students that validated this research, in which they were not used to deal with usability in early iterations. Also, more

usability mechanisms should be studied and modeled, particularly related to constructing software for modern technologies such as web and mobile.

The results presented in this research are therefore a starting contribution to formalize the integration of usability mechanisms into agile requirements. It experiments its final input to the whole development process and the final level of quality of use of the software products. The elements of the framework provided in this research have been proven to bring benefits to the final product. Still, they should be considered as only a part of the whole definition of a process to improve usability in software developed using agile methodologies.

12. ACKNOWLEDGEMENTS

I would like to express my gratitude to Dr. Ana María Moreno for her valuable guidance and feedback during the development of this research. I would also like to thank Prof. Agustin Yagüe for his assistance and insights through the process.

I would also like to express my gratitude to my parents and my two sisters for their love and support and for always making me feel proud. On the same way I thank the rest of my family for always being there for me.

Finally, I would like to thank all my friends, the ones from my country and the ones I have met during these last two years abroad, because they have been a huge support through all this experience.

13. REFERENCES

[1] T. Jokela, P. Abrahamsson. "Usability Assessment of an Extreme Programming Project. Close Co-operation with the Customer Does Not Equal Good Usability". *PROFES*, 2004.

[2] L. Constantine, L. Lockwood. "Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design". Addison-Wesley Professional, 1999.

[3] IEEE. "IEEE Std 1061: Standard for a Software Quality Metrics Methodology". 1998.

[4] ISO/IEC. "ISO 9126: Information Technology - Software quality characteristics and metrics". 1991.

[5] B. Boehm et al. "Characteristics of Software Quality". North Holland, New York, 1978.

[6] D. Fox, J. Sillito, F. Maurer. "Agile methods and user-centered design: How these two methodologies are being successfully integrated in industry". *AGILE '08. Conference*, pp. 63 -72, 2008.

[7] T. Silva da Silva, A. Martin, F. Maurer, M. Silveira, J. Ferreira. "A Systematic Literature Review on Agile Methods and User-Centered Design". *Information and Software Technology*, 2012.

[8] J. Ferreira, H. Sharp, H. Robinson. "Values and assumptions shaping agile development and user experience design in practice". *XP 2010*, pp. 178-183, 2010.

[9] A. Seffah, E. Metzker. "The obstacles and myths of usability and software engineering". *Communications of the ACM,* vol.47, no. 12, pp. 71–76, 2004.

[10] E. Folmer, J. Group, J. Bosch. "Architecting for usability: a survey". *Journal of Systems and Software,* vol. 70, pp. 61–78, 2004.

[11] D.J. Mayhew. "The Usability Engineering Lifecycle". *Morgan Kaufmann*, 1999.

[12] ISO. "ISO 18529, 00: Human-Centered Lifecyle Process Descriptions". 2000.

[13] N. Juristo, A. Moreno, M-I. Sanchez-Segura. "Guidelines for eliciting usability functionalities". *IEEE Transactions on Software Engineering*, vol. 33, no. 11, pp. 744-758, 2007.

[14] N. Juristo, A.M. Moreno, M.I. Sánchez-Segura. "Analysing the Impact on Usability on Software Design". *Journal of Systems and Software,* vol. 80, no. 9, pp. 1506-1516, 2007.

[15] M. Cohn. “User Stories Applied for Agile Software Development”. *Addison-Wesley Professional*, 1 edition, 2004.

[16] A. Moreno, A. Seffah, R. Capilla, M-I Sánchez-Segura. “HCI Practices for Building Usable Software”. *Computer*, vol. 46, no. 4, pp. 100-102, 2013.

[17] J. Patton. “Hitting the Target: Adding Interaction Design to Agile Software Development”. *OPSLA'04 Proceedings*, 2004.

[18] J. Haikara. “Usability in agile software development: Extending the interaction design process with personas approach”. *XP 2007*, pp. 153-156, 2007.

[19] N. M. Stephanie Chamberlain, H. Sharp, N. Maiden. “Towards a Framework for Integrating Agile Development and User-Centred Design”. *Springer Berlin*, 2006.

[20] B. Shneiderman. “Designing the User Interface: Strategies for Effective Human-Computer Interaction”. *Addison-Wesley*, 1998.

[21] J. Tidwell. “Designing Interfaces. Patterns for Effective Interaction Design”. *O’Reilly*, 2005.

[22] J. Nielsen. “Usability Engineering”. *John Wiley & Sons*, 1993.

[23] R. Rubinstein, H. Hersh. “The Human Factor”. *Digital Press, Bedford, MA*, 1984.

[24] L. Carvajal, A. Moreno, M.I. Sánchez-Segura, A. Seffah. “Usability through Sofware Desing”. *IEEE Transactions on Software Engineering*. Accepted for publication 2014.

[25] M. Düchting, D. Zimmermann, K. Nebe. “Incorporating user centered requirement engineering into agile software development”. *Proceedings of the 12th international conference on Human-computer interaction: interaction design and usability, HCI'07, Springer-Verlag,* Berlin, Heidelberg, pp. 58-67, 2007.

[26] M. Singh. “U-SCRUM: An agile methodology for promoting usability”. *Proceedings of the Agile 2008, IEEE Computer Society, Washington, DC, USA*, pp. 555-560, 2008.

[27] H. Beyer, K. Holtzblatt, L. Baker. “An agile customer-centered method: Rapid contextual design”. *Extreme Programming and Agile Methods - XP/Agile Universe 2004*, pp. 527-554, 2004.

[28] R. Carbon, J. Dörr, M. Trapp. “Focusing extreme programming on usability”. *P. Dadam, M. Reichert (Eds.), GI Jahrestagung (2)*, vol. 51, pp. 147-152, 2004.

[29] Trello. URL: http://www.trello.com/, visited on July, 2013.

[30] A. Moreno, A. Yagüe, D. Yucra. "Tailoring user stories to deal with usability". *Jornada de Ingeniería del Software y Bases de Datos*, 2012.

[31] L. Fuentes-Fernandez, A. Vallecillo-Moreno. "An Introduction to UML Profiles". *UPGRADE The European Journal for the Informatics Professional*, vol. V, no. 2, pp. 5-13, 2004.

[32] Agilefant. URL: http://www.agilefant.org/, visited on January, 2013.

[33] IceScrum. URL: http://www.icescrum.org/, visited on January, 2013.

[34] Kunagi. URL: http://www.kunagi.org/, visited on January, 2013.

[35] A. Santos. "Biblioideas". *Master Thesis, Universidad Politécnica de Madrid*, 2013.

APPENDIX A. OCL CONSTRAINT MODEL FOR THE UML PROFILE.

```
context UML::InfrastructureLibrary::Core::Constructs::Class

	- Definition of allowed relations of each stereotype

	inv :

	self.isStereotyped("Usability Mechanism")

		implies

		self.connection->select(isStereotyped("affects"))->size > 0 and

		self.connection->select(isStereotyped("creates"))->size >= 0 and

		self.connection->select(isStereotyped("modifies"))->size >= 0 and

		self.connection->reject(isStereotyped("affects") or
isStereotyped("creates") or isStereotyped("modifies"))->isEmpty

	inv :

	self.isStereotyped("User Story")

		implies

		self.connection->select(isStereotyped("affects"))->size >= 0 and

		self.connection->select(isStereotyped("creates"))->size <= 1 and

		self.connection->select(isStereotyped("modifies"))->isEmpty and

		self.connection->select(isStereotyped("relates to"))->size >= 0

	inv :

	self.isStereotyped("Task")

		implies

		self.connection->select(isStereotyped("affects"))->size >= 0 and

		self.connection->select(isStereotyped("creates"))->isEmpty  and

		self.connection->select(isStereotyped("modifies"))->isEmpty and

		self.connection->select(isStereotyped("relates to"))->isEmpty

	inv :

	self.isStereotyped("Acceptance Criteria")
```

```
            implies
            self.connection->select(isStereotyped("affects"))->isEmpty and
            self.connection->select(isStereotyped("creates"))->size <= 1 and
            self.connection->select(isStereotyped("modifies"))->size <= 1 and
            self.connection->select(isStereotyped("relates to"))->isEmpty
      inv :
      self.isStereotyped("Usability User Story")
            implies
            self.connection->select(isStereotyped("creates"))->size = 1 and
            self.connection->select(isStereotyped("relates to"))->size > 1
and
            self.connection->reject(isStereotyped("creates") or
isStereotyped("relates to"))->isEmpty
      inv :
      self.isStereotyped("Independent Usability Task")
            implies
            self.connection->select(isStereotyped("creates"))->size = 1 and
            self.connection->reject(isStereotyped("creates"))->isEmpty
      inv :
      self.isStereotyped("Dependent Usability Task")
            implies
            self.connection->select(isStereotyped("affects"))->size > 1 and
            self.connection->select(isStereotyped("creates"))->size = 1 and
            self.connection->reject(isStereotyped("affects") or
isStereotyped("creates"))->isEmpty
context UML::InfrastructureLibrary::Core::Constructs::Association
      - Definition of allowed participant stereotypes in each relation
      inv : self.isStereotyped("affects")
            implies
```

```
            self.connection->reject(c1, c2 |
(c1.participant.isStereotyped("Usability Mechanism") and

            c2.participant.isStereotyped("User Story")) or
(c1.participant.isStereotyped("Task") and

            c2.participant.isStereotyped("Dependent Usability Task")))-
>isEmpty

      inv : self.isStereotyped("creates")

            implies

            self.connection->reject(c1, c2 |
(c1.participant.isStereotyped("Usability Mechanism") and
c2.participant.isStereotyped("User Story")) or

            (c1.participant.isStereotyped("Usability Mechanism") and
c2.participant.isStereotyped("Acceptance Criteria")) or

            (c1.participant.isStereotyped("Usability Mechanism") and
c2.participant.isStereotyped("Usability User Story)) or

            (c1.participant.isStereotyped("Usability Mechanism") and
c2.participant.isStereotyped("Independent Usability Task")) or

            (c1.participant.isStereotyped("Usability Mechanism") and
c2.participant.isStereotyped("Dependent Usability Task")))->isEmpty

      inv : self.isStereotyped("modifies")

            implies

            self.connection->reject(c1, c2 |
c1.participant.isStereotyped("Usability Mechanism") and

            c2.participant.isStereotyped("Acceptance Criteria"))->isEmpty

      inv : self.isStereotyped("relates to")

            implies

            self.connection->reject(c1, c2 |
c1.participant.isStereotyped("User Story") and
c2.participant.isStereotyped("Usability User Story"))->isEmpty

      inv : self.connection->exists(participant.isStereotyped("User Story"))
and (not (self.isStereotyped("affects") or self.isStereotyped("creates") or


self.isStereotyped("modifies") or self.isStereotyped("relates to"))
```

```
            implies

            ((self.connection->exists(participant.isStereotyped("Task")) and
multiplicity.min = 1) or

            (self.connection->exists(participant.isStereotyped("Acceptance
Criteria")) and

            multiplicity.min = 1)) and (self.connection-
>reject(participant.isStereotyped("Task") or

            participant.isStereotyped("Acceptance Criteria"))->isEmpty)

      inv : self.connection->exists(participant.isStereotyped("Task")) and
(not (self.isStereotyped("affects") or self.isStereotyped("creates") or


self.isStereotyped("modifies") or self.isStereotyped("relates to"))

            implies

            (self.connection->exists(participant.isStereotyped("User Story"))
and multiplicity.min = 1 and multiplicity.max = 1)

            and (self.connection->reject(participant.isStereotyped("User
Story"))->isEmpty)

      inv : self.connection->exists(participant.isStereotyped("Acceptance
Criteria")) and (not (self.isStereotyped("affects") or
self.isStereotyped("creates") or


self.isStereotyped("modifies") or self.isStereotyped("relates to"))

            implies

            (self.connection->exists(participant.isStereotyped("User Story"))
and multiplicity.min = 1 and multiplicity.max = 1)

            and (self.connection->reject(participant.isStereotyped("User
Story"))->isEmpty)
```

APPENDIX B. USABILITY MECHANISM MODELS.

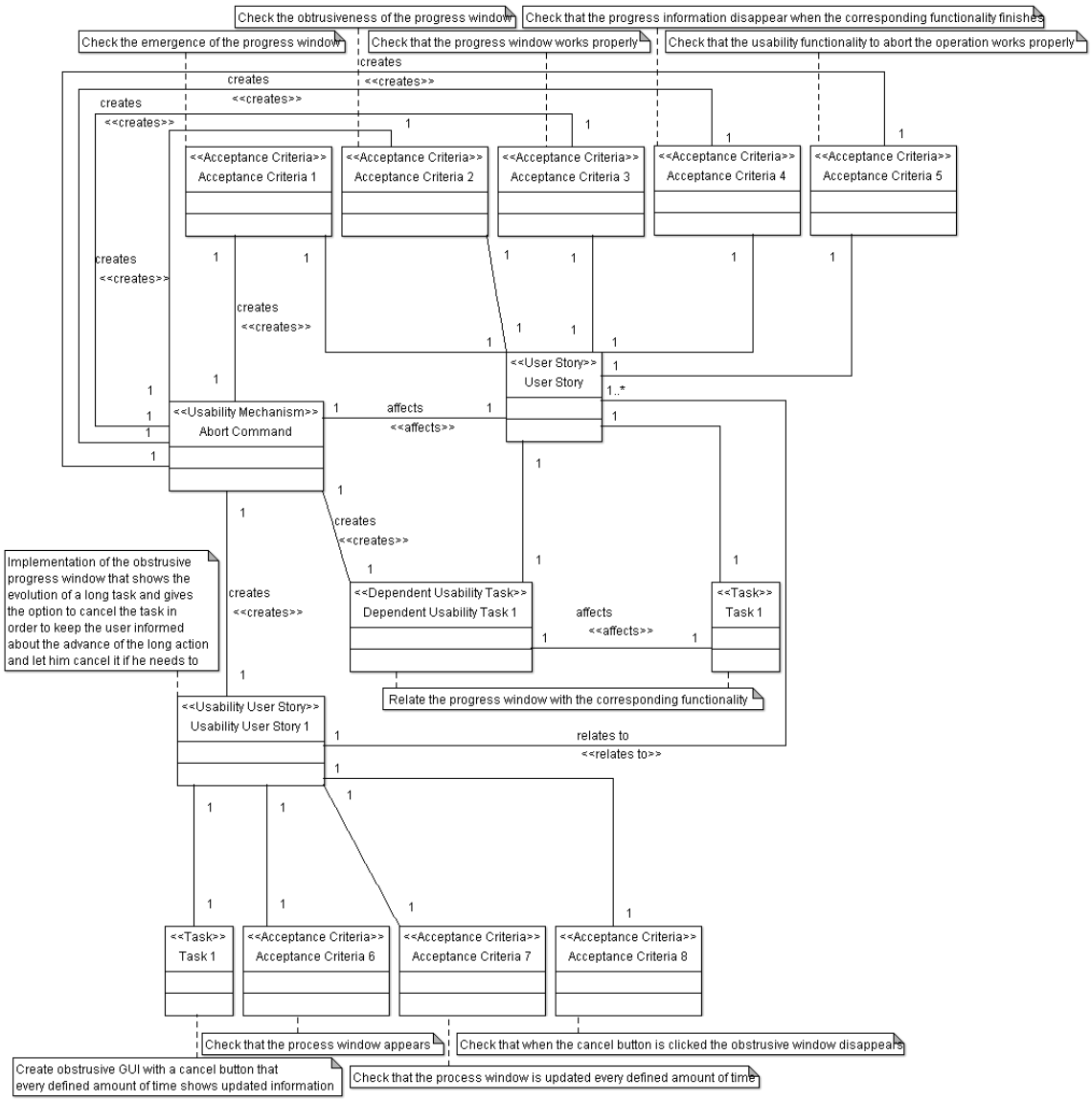

FIGURE 15. ABORT COMMAND MODEL

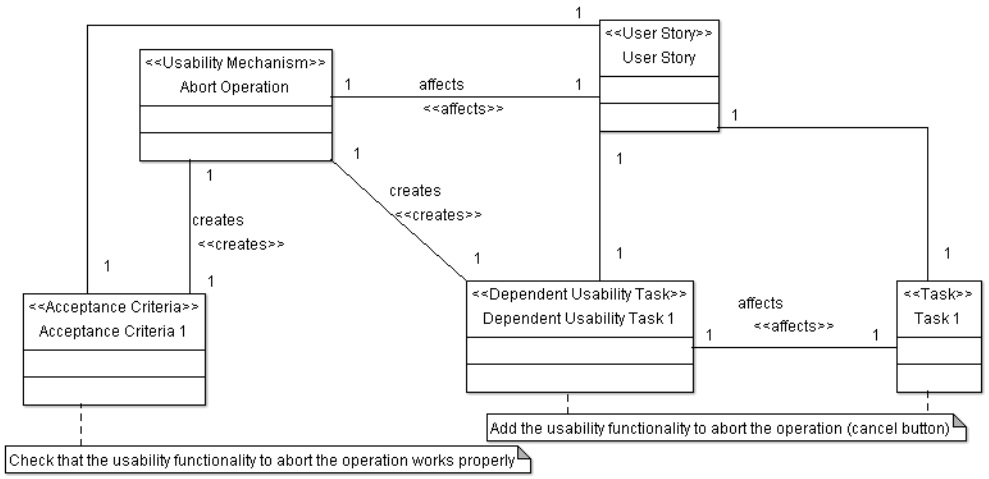

FIGURE 16. ABORT OPERATION MODEL

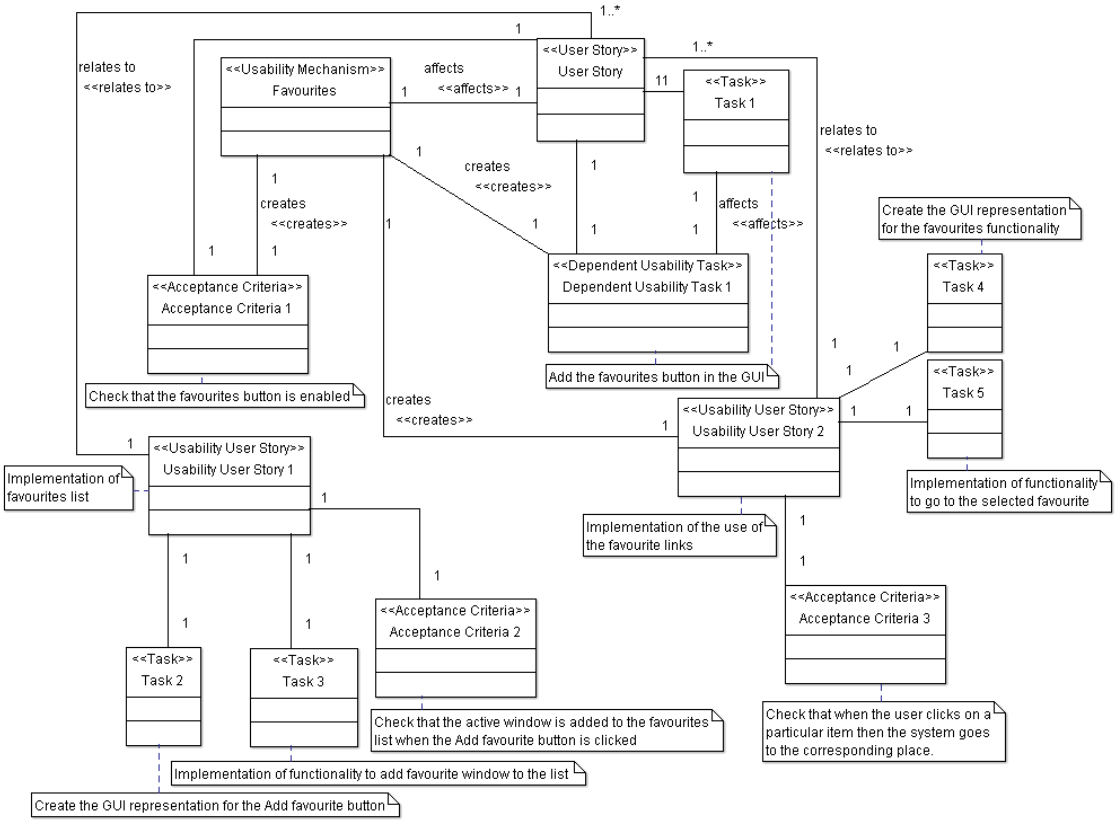

FIGURE 17. FAVOURTIES MODEL

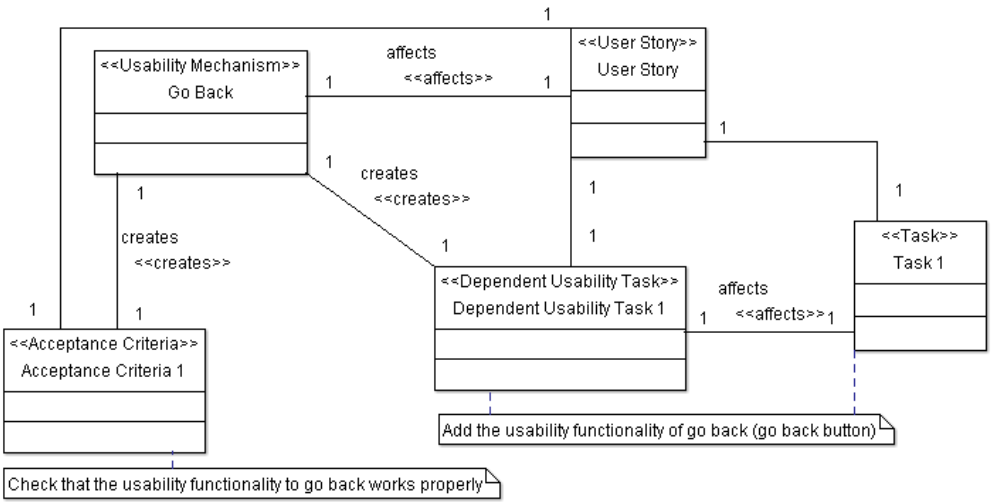

FIGURE 18. GO BACK MODEL

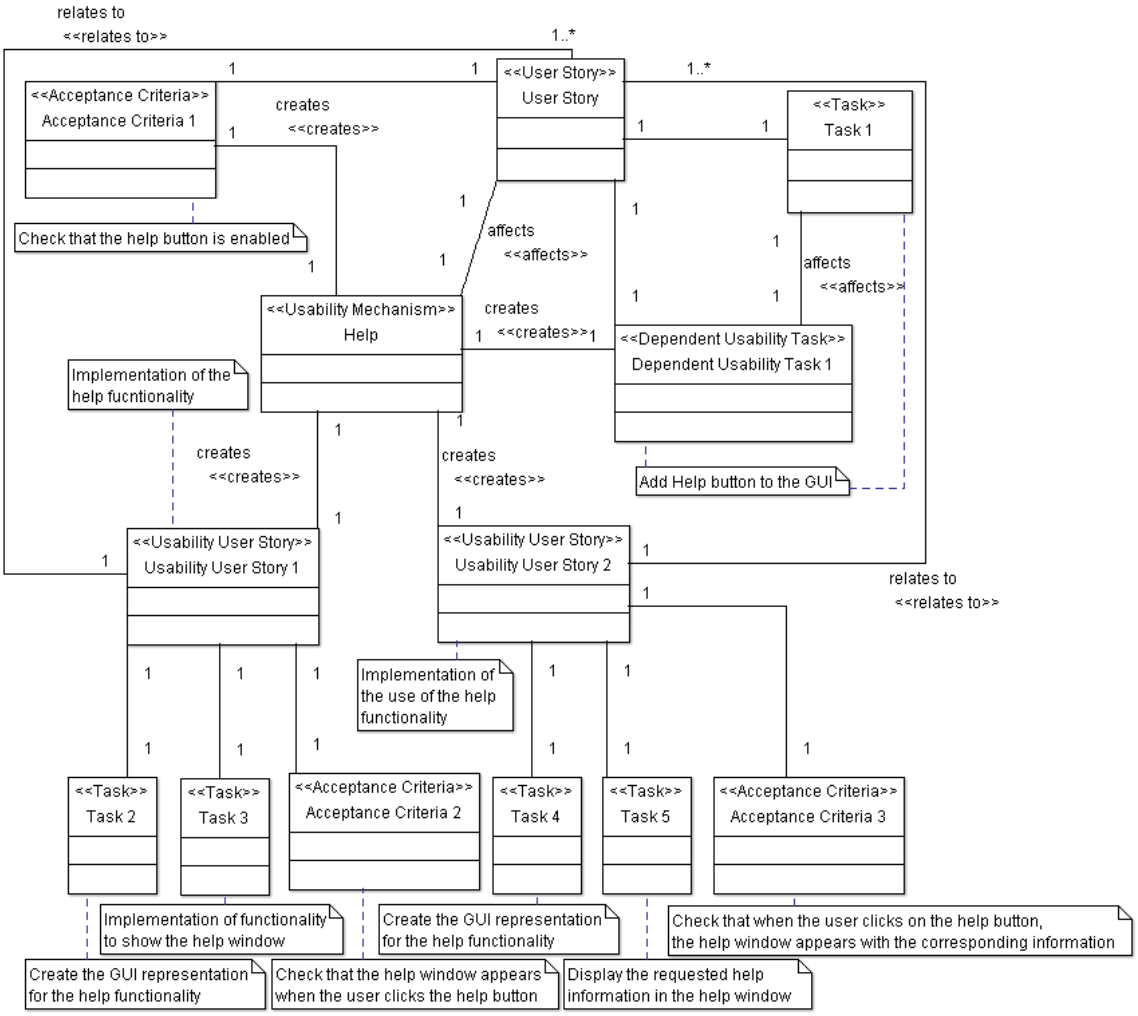

FIGURE 19. HELP MODEL

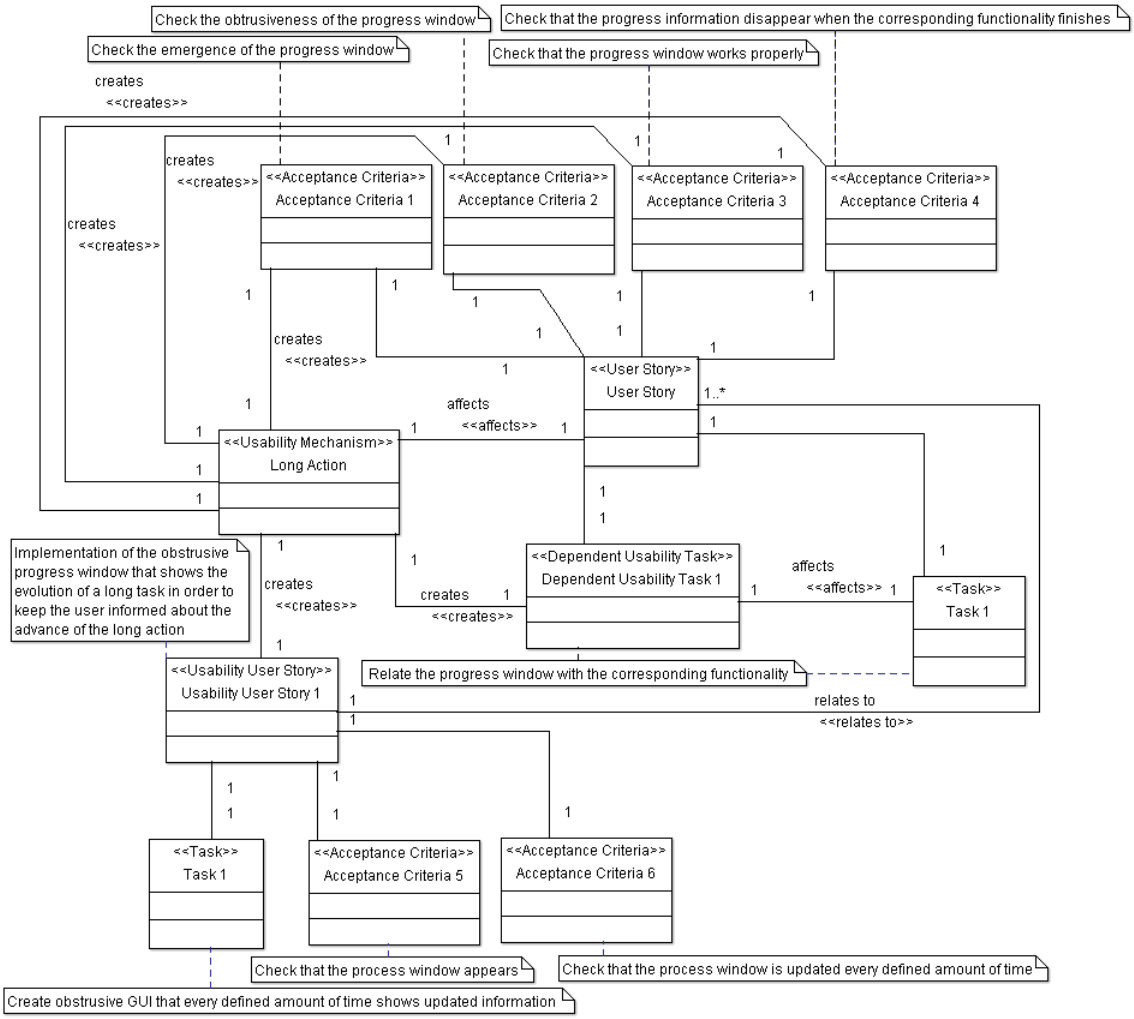

FIGURE 20. LONG ACTION MODEL

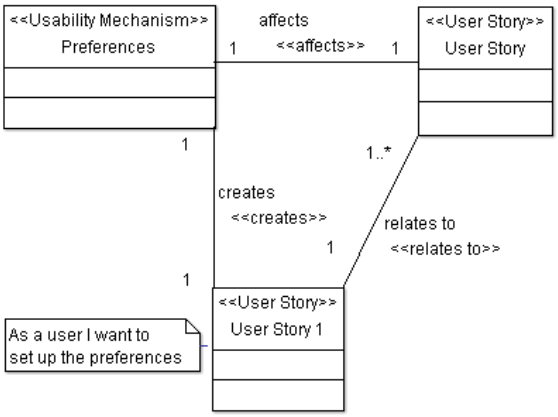

FIGURE 21. PREFERENCES MODEL

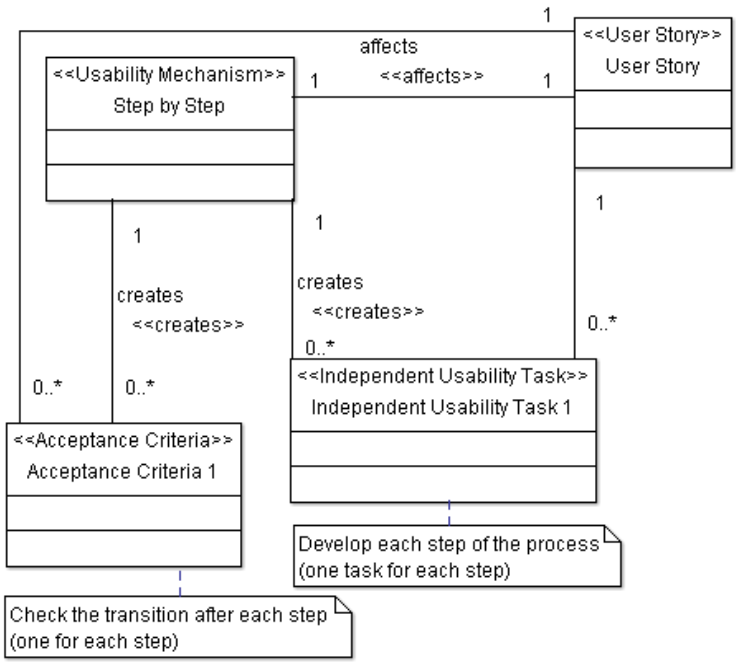

FIGURE 22. STEP BY STEP MODEL

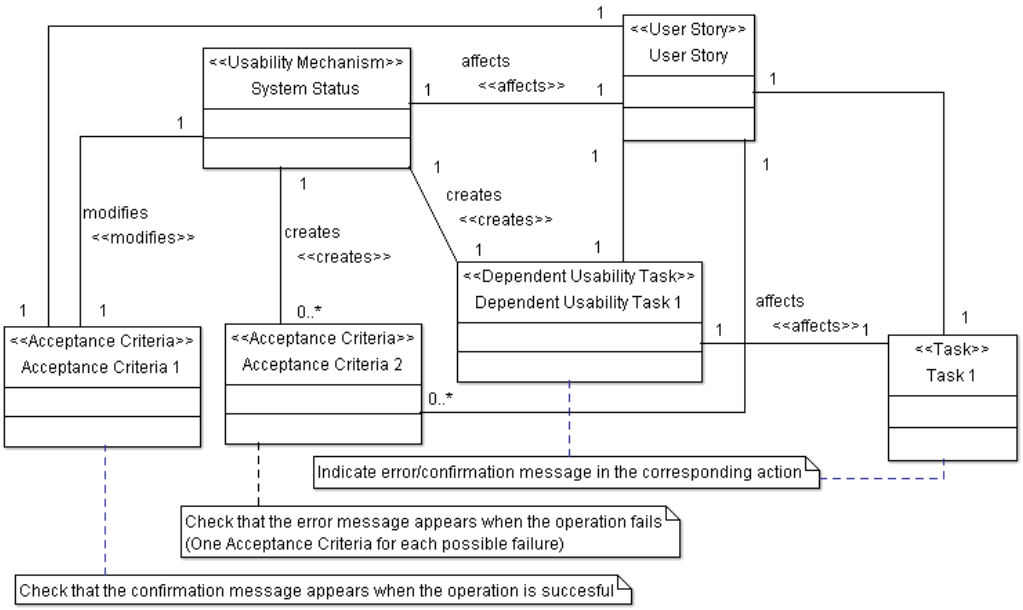

FIGURE 23. SYSTEM STATUS MODEL

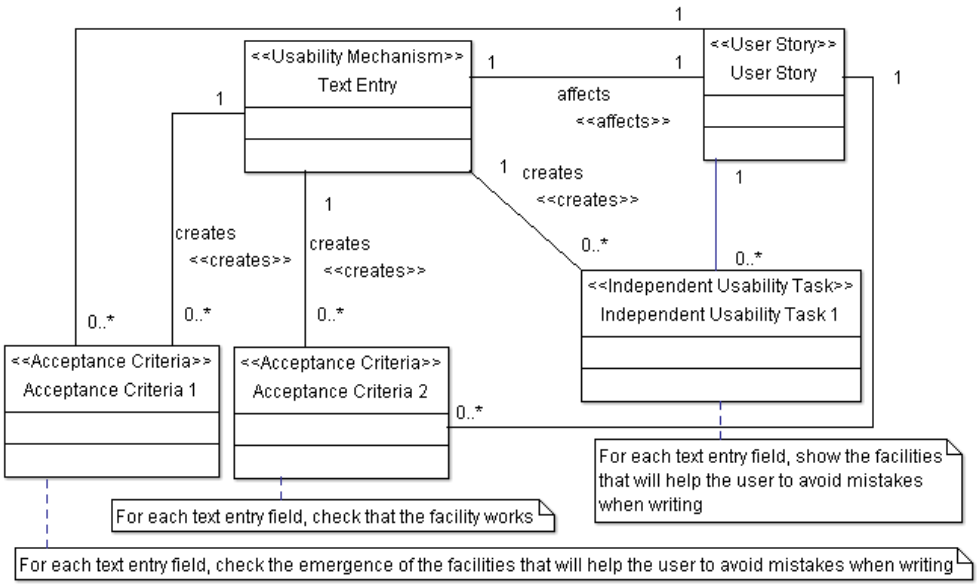

FIGURE 24.TEXT ENTRY MODEL

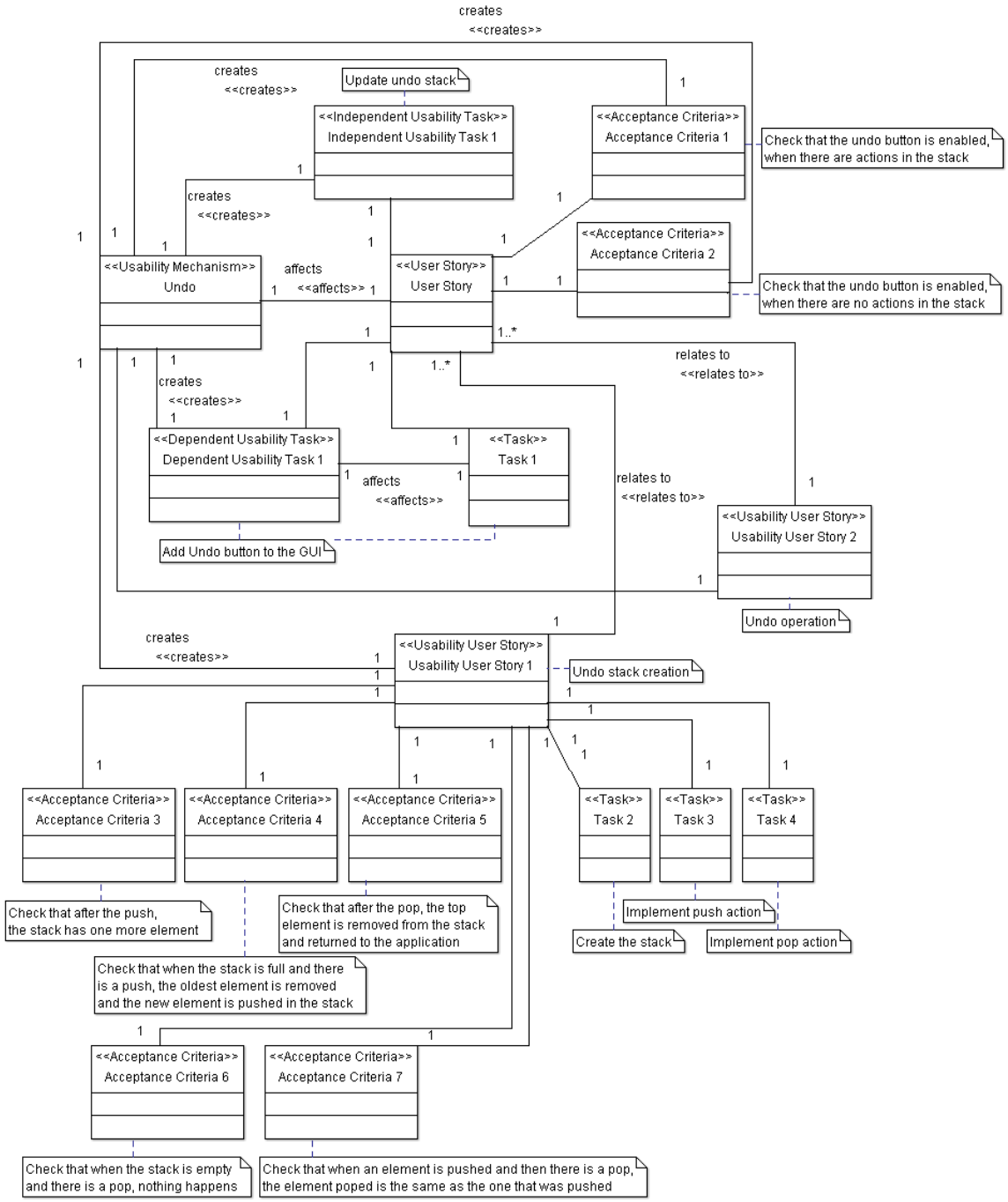

FIGURE 25. UNDO MODEL

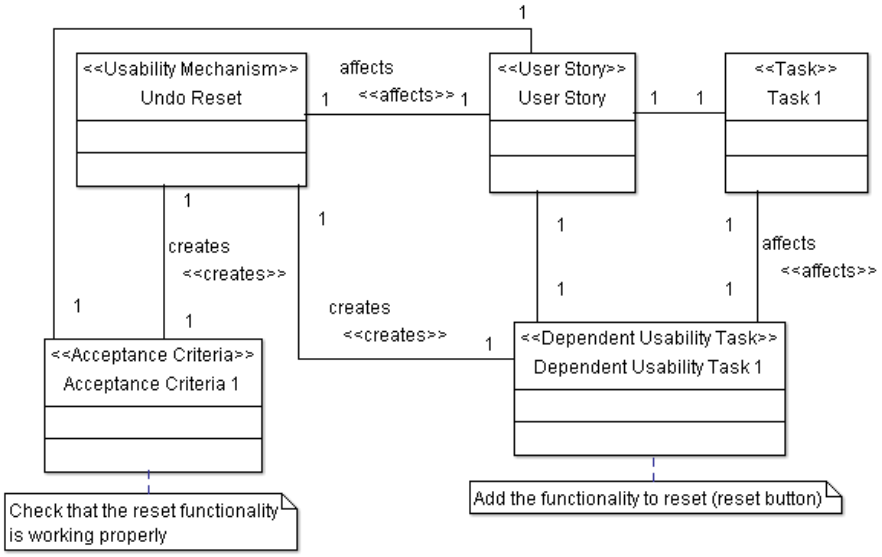

FIGURE 26. UNDO RESET MODEL

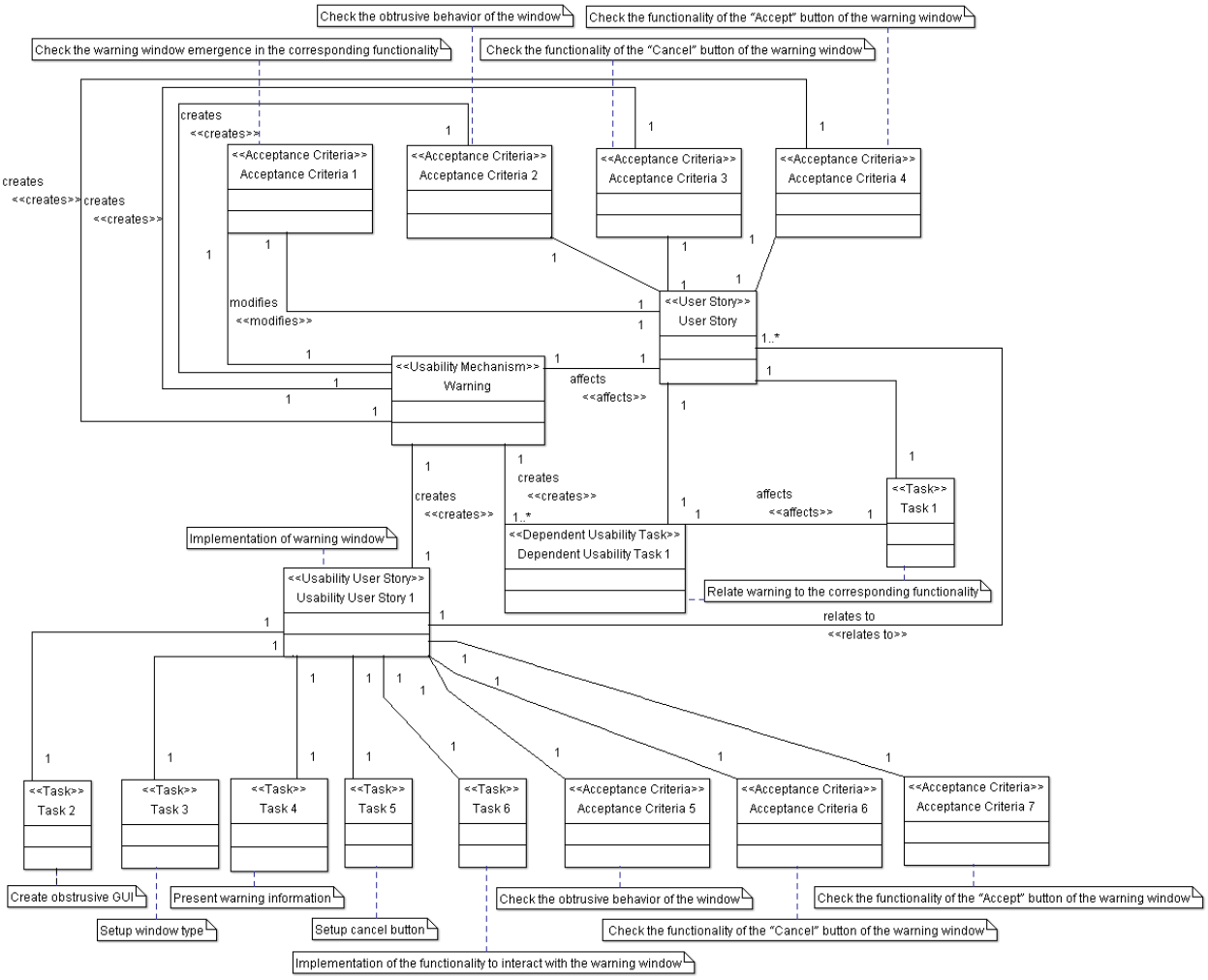

FIGURE 27. WARNING MODEL

APPENDIX C. QUESTIONNAIRE FOR VALIDATION.

1. With what frequency you used Usability Mechanisms in your user stories?
    - In none of the user stories
    - In less than a half of the user stories
    - In more than a half of the user stories
    - In practically all user stories
2. Enumerate from 1 to 13 the Usability Mechanisms ordering them by the frequency on which you used them on the user stories, where 1 means that it is the least used and 13 the one you used the most (Leave blank all the ones you didn't use):

_ System Status	_ Text Entry
_ Warning	_ Undo
_ Long Action	_ Undo Reset
_ Long Action + Abort Command	_ Step by Step
	_ Preferences
_ Abort Operation	_ Favorites
_ Go Back	_ Help

3. What percentage of the effort did the extra work done by the addition of Usability Mechanisms meant?
    - Less than 25%
    - Between 26% and 50%
    - Between 51% y 75%
    - More than 75%
4. What kind of extra work?

---

5. Do you think that the using Usability Mechanisms from the beginning of the development saved you extra work in the following iterations?
    - Yes
    - No
6. What kind of extra work?

---

7. What percentage of the usability functionality of your final product do you think you would have added if you didn't use Usability Mechanisms?
    - Less than 25%

- Between 26% and 50%
- Between 51% y 75%
- More than 75%

8. How would you rate the usability of your final product? (Where 1 would be a rating for low or no usability and 5 for very high usability)
    - 5
    - 4
    - 3
    - 2
    - 1
9. Do you think this experience, of using Usability Mechanisms, will help you think about usability from the beginning of the development process even if you don't have a tool that explicitly helps you in the addition of usability functionality?
    - Yes
    - No
10. How would you rate the usability of the tool, U-Kunagi, specifically on the management of Usability Mechanisms inside user stories? (Where 1 would be a rating for low or no usability and 5 for very high usability)
    - 5
    - 4
    - 3
    - 2
    - 1
11. Why?

---

12. What is your opinion on using Usability Mechanisms and their respective recommendations when working with user stories?
